**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ

“___” червня 2020 р.

# Дипломний проект

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою «Системне програмування»**

зі спеціальності **123 «Комп'ютерна інженерія»**

на тему: G-модель багатопроцесорної системи, процесори якої мають різні можливості з реконфігурування

Виконав: студент IV курсу, групи КВ-61
(шифр групи)

Федорченко Денис Олегович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник проф. каф. СПіСКС, д. т. н. Романкевич О.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М.
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали ) (підпис)

Рецензент_____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020

# ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

| № з/п | Формат | Позначення | Найменування | Кількість листів | Примітка |
|---|---|---|---|---|---|
| 1 | А4 | | Завдання на дипломний проект | 2 | |
| 2 | А4 | ДП 045450.004 ПЗ | Пояснювальна записка | 58 | |
| 3 | А4 | ДП 045450.005 Д1 | Структура взаємодії класів додатку. Схема структурна. | 1 | |
| 4 | А4 | ДП 045450.006 Д2 | Генерація таблиці K(m, n). Схема алгоритму. | 1 | |
| 5 | А4 | ДП 045450.007 Д2 | Алгоритм склейки булевих функцій. Схема алгоритму. | 1 | |
| 6 | А4 | ДП 045450.008 Д2 | Генерація внутрішніх ребер. Схема алгоритму. | 1 | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| | | | | ДП 045450 00.003 ТП | | |
|---|---|---|---|---|---|---|
| | ПІБ | Підп. | Дата | | | |
| Розробн. | Федорченко Д.О | | | Відомість дипломного проекту | Лист | Листів |
| Керівн. | Романкевич О.М. | | | | 1 | 1 |
| Консульт. | | | | | КПІ ім. Ігоря Сікорського Каф. СПіСКС Гр. КВ-61 | |
| Н/контр. | Клятченко Я.М. | | | | | |
| Зав.каф. | Романкевич В.О. | | | | | |

# Пояснювальна записка
## до дипломного проекту

на тему: G-модель багатопроцесорної системи, процесори якої мають різні можливості з реконфігурування

Київ – 2020

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____20___ р.

**ЗАВДАННЯ**
**на дипломний проєкт студента**
Федорченку Денису Олеговичу

1. Тема проєкту «G-модель багатопроцесорної системи, процесори якої мають різні можливості з реконфігурування»,
керівник проєкту проф. каф. СПіСКС, д. т. н. Романкевич О.М., затверджені наказом по університету від «__» _____ 20__ р. №_____

2. Термін подання студентом проєкту 22.05.2020

3. Вихідні дані до проєкту Назва. G-модель багатопроцесорної системи, процесори якої мають різні можливості з реконфігурування.

4. Зміст пояснювальної записки

1. Елементи теорії надійності
2. Огляд методів моделювання надійності
3. Графо-логічна модель
4. Огляд реалізації додатку

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) презентація; структурна схема взаємодії класів

додатку; схема алгоритму генерації таблиці K(m, n); схема алгоритму склейки булевих функцій; схема алгоритму генерації внутрішніх ребр.

6. Консультанти розділів проєкту*

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| нормоконтроль | Клятченко Я.М., доц.каф.СПСКС, к.т.н. | | |
| | | | |

7. Дата видачі завдання 18.10.2019.

Календарний план

| № з/п | Назва етапів виконання дипломного проєкту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1. | Вивчення літератури за тематикою проєкту | 18.11.2019 | |
| 2. | Розроблення та узгодження технічного завдання | 30.11.2019 | |
| 3. | Аналіз існуючих рішень | 10.01.2020 | |
| 4. | Підготовка матеріалів першого розділу дипломного проєкту | 18.01.2020 | |
| 5. | Підготовка матеріалів другого розділу дипломного проєкту | 16.02.2020 | |
| 6. | Підготовка матеріалів третього розділу дипломного проєкту | 14.03.2020 | |
| 7. | Підготовка матеріалів четвертого розділу дипломного проєкту | 01.04.2020 | |
| 8. | Підготовка графічної частини дипломного проєкту | 01.05.2020 | |
| 9. | Оформлення документації дипломного проєкту | 14.05.2020 | |
| 10. | Попередній огляд матеріалів диплому на кафедрі | 20.05.2020 | |

Студент _____ Денис ФЕДОРЧЕНКО
(підпис)

Керівник проєкту _____ Олексій РОМАНКЕВИЧ
(підпис)

# АНОТАЦІЯ

Метою даного дипломного проєкту є розробка програмного забезпечення для побудови та модифікації базових графо-логічних моделей. Проєкт дозволяє швидше будувати такі моделі, вирішуючи задачі пошуку реберних функцій та їх мінімізації.

Для вирішення задачі була розроблена модульна структура, що включає модулі для обробки вводу користувача, роботи з векторною графікою для відображення моделі на екран, генерацію булевих функцій довільної складності, модуль для перетворення об’єктного представлення функцій в формі, сприйнятній для людини, модулі для генерації та мінімізації реберних функцій, обрахунку зв’язності та інші.

В ході роботи над дипломним проєктом було створено додаток, що має такі можливості:

– кросплатформність, тобто робота на Windows, Mac та Linux;

– простий в користуванні графічний інтерфейс;

– побудова базової моделі з заданими кількістю змінних та степенем відмовостійкості;

– мінімізація моделі для спрощення структури графу;

– перевірка реакції моделі на довільні вектори відмов;

– модифікація моделі для забезпечення стійкості моделі до довільного вектору відмови;

Ключові слова: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ЗАСОБИ ОЦІНКИ НАДІЙНОСТІ, GL-МОДЕЛЬ, ГРАФОВІ МОДЕЛІ НАДІЙНОСТІ, ДЕСКТОПНИЙ ДОДАТОК.

## ABSTRACT

The purpose of this thesis is to develop software for building and modifying basic graph-logical models. The project allows you to build such models faster, solving the problem of finding edge functions and minimizing them.

To solve the problem, a modular structure was developed, including modules for processing user input, working with vector graphics to display the model on the screen, generating Boolean functions of arbitrary complexity, a module for converting object representations of functions into human-friendly form, modules for generating and minimization of costal functions, calculation of connectivity and others.

In the course of work on the diploma project, an application was created that has the following features:

- cross-platform, ie work on Windows, Mac and Linux;
- easy-to-use graphical interface;
- construction of a basic model with a given number of variables and the degree of fault tolerance;
- minimization of the model to simplify the structure of the graph;
- check the reaction of the model to arbitrary failure vectors;
- modification of the model to ensure the stability of the model to an arbitrary failure vector;



Keywords: SOFTWARE, RELIABILITY ASSESSMENT TOOLS, GL-MODEL, GRAPHIC RELIABILITY MODELS, DESKTOP APPENDIX.

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | А4 | ІАЛЦ.045450.002 ТЗ | G-модель | 4 | | |
| | | | багатопроцесорної системи | | | |
| | | | процесори якої мають | | | |
| | | | різні можливості з | | | |
| | | | реконфігурування | | | |
| | | | Технічне завдання | | | |
| | А4 | ІАЛЦ.045450.003 ТП | G-модель | 2 | | |
| | | | багатопроцесорної системи | | | |
| | | | процесори якої мають | | | |
| | | | різні можливості з | | | |
| | | | реконфігурування | | | |
| | | | Відомості технічного | | | |
| | | | проєкту | | | |
| | А4 | ІАЛЦ.045450.004 ПЗ | G-модель | 58 | | |
| | | | багатопроцесорної системи | | | |
| | | | процесори якої мають | | | |
| | | | різні можливості з | | | |
| | | | реконфігурування | | | |
| | | | Пояснювальна записка | | | |
| | | | | | | |
| | | | | | | |

| Змін. | Арк. | № докум. | Підпис | Дата | ІАЛЦ.045450.001 ОА | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Федорченко Д.О | | | G-модель багатопроцесорної системи, процесори якої мають різні можливості з реконфігурування<br>**Опис альбому** | Літ. | Аркуш 1 / Аркушів 2 |
| Перевірив | | Романкевич ОМ | | | | | |
| Консульт. | | | | | | КПІ ім. Ігоря Сікорського, ФПМ КВ-61 | |
| Н. контроль | | Клятченко Я.М. | | | | | |
| Зав. каф. | | Романкевич В.О | | | | | |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | А4 | ІАЛЦ.045450.005 Д1 | Структура взаємодії | 1 | | |
| | | | класів додатку. | | | |
| | | | Структурна схема. | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045450.006 Д2 | Генерація таблиці K(m, n). | 1 | | |
| | | | Схема алгоритму. | | | |
| | | | | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045450.007 Д2 | Алгоритм склейки | 1 | | |
| | | | булевих функцій. | | | |
| | | | Схема алгоритму. | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045450.008 Д2 | Генерація внутрішніх | 1 | | |
| | | | ребер. | | | |
| | | | Схема алгоритму. | | | |
| | | | | | | |
| | | | | | | |
| | | Диск CD-ROM | Текст пояснювальної | 1 | | |
| | | | записки. | | | |
| | | | Графічні матеріали. | | | |
| | | | | | | |
| | | | | | | |

# ЗМІСТ

## 1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «G-модель багатопроцесорної системи, процесори якої мають різні можливості з реконфігурування».

Галузь застосування: розробка користувацького десктопного додатку для спрощення та автоматизації побудови GL-моделей оцінки надійсноті.

## 2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

## 3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є створення додатку для побудови та модифікацій базових GL-моделей.

## 4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації в періодичних виданнях та електронні статті у мережі Інтернет.

## 5. ТЕХНІЧНІ ВИМОГИ

1.1 Вимоги до програмного продукту, що розробляється

- побудова базових k-out-of-n моделей;
- мінімізація моделі;
- простий графічний інтерфейс;
- вивід логічних функцій;

- проведення внутрішніх ребер;
- кросплатформенність.

5.2 Вимоги до апаратного забезпечення

- оперативна пам’ять: 4 Гб.

## 6. ЕТАПИ РОЗРОБКИ

| № з/п | Назва етапів виконання дипломного проєкту | Термін виконання етапів |
|---|---|---|
| 1. | Видача завдання на дипломне проєктування | 18.10.2019 |
| 2. | Вивчення літератури за тематикою роботи | 18.11.2019 |
| 3. | Розроблення та узгодження технічного завдання | 30.11.2019 |
| 4. | Розроблення структури додатку | 16.01.2020 |
| 5. | Розроблення дизайну та графічних елементів | 10.02.2020 |
| 6. | Програмна реалізація додатку | 16.02.2020 |
| 7. | Тестування додатку | 14.03.2020 |
| 8. | Підготовка матеріалів текстової частини проєкту | 01.04.2020 |
| 9. | Підготовка матеріалів графічної частини проєкту | 01.05.2020 |
| 10. | Оформлення технічної документації проєкту | 14.05.2020 |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | А4 | ІАЛЦ.045450.004 ПЗ | G-модель | 58 | | |
| | | | багатопроцесорної системи | | | |
| | | | процесори якої мають | | | |
| | | | різні можливості з | | | |
| | | | реконфігурування | | | |
| | | | Пояснювальна записка | | | |
| | А4 | ІАЛЦ.045450.005 Д1 | Структура взаємодії | 1 | | |
| | | | класів додатку. | | | |
| | | | Структурна схема. | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045450.006 Д2 | Алгоритм роботи | 1 | | |
| | | | смарт-контракту. | | | |
| | | | Схема алгоритму. | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045450.007 Д2 | Генерація таблиці K(m, n). | 1 | | |
| | | | Схема алгоритму. | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

| Змін. | Арк. | № докум. | Підпис | Дата | ІАЛЦ.045450.003 ТП | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Федорченко Д.О | | | G-модель багатопроцесорної системи, процесори якої мають різні можливості з реконфігурування<br>**Відомість технічного проекту** | Літ. | Аркуш 1 / Аркушів 1 |
| Перевірив | | Романкевич ОМ | | | | | |
| Консульт. | | | | | | КПІ ім. Ігоря Сікорського, ФПМ КВ-61 | |
| Н. контроль | | Клятченко Я.М. | | | | | |
| Зав. каф. | | Романкевич В.О | | | | | |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | А4 | ІАЛЦ.045450.008 Д2 | Генерація внутрішніх | 1 | | |
| | | | ребер. | | | |
| | | | Схема алгоритму. | | | |

| Змін. | Арк. | № докум. | Підпис | Дата | ІАЛЦ.045450.003 ТП | Арк. 2 |
|---|---|---|---|---|---|---|

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ

АДВ – аналіз дерева відмов.

БСН – блок-схема надійності.

ВМС – відмовостійка мультипроцесорна система.

GL-модель – графо-логічна модель.

ДВ – дерево відмов.

ДНФ – диз’юнктивна нормальна форма.

ФВ – фізика відмов.

API - application programming interface.

CSS – cascade stylesheet.

GUI – graphical user interface.

JS – JavaScript.

HTML – Hypertext markup language.

# ВСТУП

Багатопроцесорність — це поєднання двох або більше центральних процесорів в одній комп'ютерній системі. Багатопроцесорні системи отримали дуже широке застосування, адже на сьогодні це основний спосіб для створення високопродуктивних комп’ютерних систем.

Першою багатопроцесорною системою є MOC ILLIAC IV, побудований в 1965 році в університеті Іллінойсу. Проєкт включав 256 процесорів, проте побудована машина мала лише 64, але в 1972 році, коли вартість системи виросла до 31 мільйона доларів, зупинились на 16. А уже в 80-х роках з'явилися перші промислові зразки мультипроцесорних систем.

Спочатку багатопроцесорні системи використовувались для задач що потребували значних ресурсів для обчислень. Зараз же, разом з ростом бізнесу, зростає кількість компаній, що широко використовують комп’ютерні технології. Багатопроцесорні системи використовуються для критично важливих застосунків, пов’язаних з обробкою транзакцій в реальному часі, підтримки систем прийняття рішень і обслуговуванням телекомунікацій.

Разом зі збільшенням сфери застосування, відповідно зростає складність задач, для яких використовують такі системи. Деяких список фундаментальних і прикладних проблем, які було б неможливо вирішити без надпотужних обчислювальних ресурсів:

- структурна біологія;
- астрономія;
- побудова напівпровідних приладів;
- наука про матеріали;
- явище надпровідності;
- фармацевтичні препарати;
- генетика;
- прогнозування погоди чи глобальних змін в атмосфері;

– квантова хромодинаміка;
– ефективність систем згоряння палива;
– гідро- і газодинаміка;
– вивчення світового океану;
– розпізнавання та синтез мови;
– геоінформаційні системи;
– розвідка земних надр;
– керований термоядерний синтез;
– транспортні задачі;
– розпізнавання зображень.

Очевидно, що з ростом сфери застосувань ростуть і вимоги до надійності багатопроцесорних систем.

Часто надійність визначають як відсутність недопустимих змін чи завад при експлуатації. Проте при зростанні складності системи значно зростає ймовірність виникнення завад та непередбачуваних ситуацій.

Системи, які опираються на безвідмовність усіх свої підсистем часто не можуть забезпечувати достатньої надійності.

Оскільки сучасні системи можуть складатися з тисячі, а то і десятків тисяч підсистем, на практиці неможливо зберегти безвідмовну роботу кожного елементу. Важливим стає характеристика відмовостійкості.

Відмовостійкість - здатність системи зберігати свою функціональність після відмови одного чи декількох компонентів. Мова йдеться саме про збереження функціональності, проте ефективність системи може поступово спадати, або ж деградувати. Тобто збільшується час відклику, система може обмежити пропускну здатність.

Будь-яка інвестиція в розробку моделі відмовостійкості системи в результаті може значно скоротити затрати на проєктування та тестування.

Чим складнішою і дорожчою в розробці та тестуванні є система, тим більш важливою є можливість виявляти слабкі місця системи ще на етапі проєктування, і тим важливішою є модель для проведення теоретичних експериментів над системою.

# 1 ТЕОРІЯ НАДІЙНОСТІ

## 1. 1 Поняття надійності

Ефективність роботи технічних систем багато в чому залежить від показнику надійності не тільки окремих пристроїв, що є модулями приладу, але і окремих елементів.

Надійність - характеристика, що показує здатність системи зберігати в встановлених межах свої властивості виконувати певні функції, враховуючи умови використання. Часто надійність визначають як відсутність недопустимих змін чи завад при експлуатації. [1]

Теорія надійності викликає значний інтерес з наступних причин:

- постійне зростання складності систем;
- уповільнення зростання надійності компонентів елементів при збільшенні числа цих елементів;
- зростання критичної важливості функцій, що виконують пристрої, а, отже, і вимог до їх надійності;
- робота систем в складніших умовах використання.

Теорія надійності вивчає:

- характеристики та критерії;
- аналіз надійності систем та методи аналізу;
- вирішення задачі синтезу системи з наперед заданою надійністю;
- підвищення надійності пристроїв на різних етапах, таких як проєктування чи експлуатація;
- тестування надійності систем.

В теорії надійності базовими є поняття системи, компонента та елемента.

Елемент – будь-який компонент системи чи окремий пристрій, певна підсистема чи будь-яке обладнання що можна розглядати як самостійну одиницю.

Елемент це будь-яка самостійна частина системи що має власну оцінку надійності і виконує свою окрему функцію в інтересах цілої системи.

Система – відповідно сукупність елементів, що взаємодіють та є взаємопов'язаними між собою.

Система будується з конкретною ціллю, що позначається у вигляді вимог до функціональності системи, певні умови експлуатації та певну ієрархічну структуру.

Системи, що вивчаються теорією надійності, можна умовно поділити на відновні, тобто такі, в яких для відновлення нормальної роботи пристрою проводять заміну несправного об'єкта після прояву відмови, та невідновні, в яких не відбувається така заміна.

Виділені чотири стани, в яких можуть перебувати елементи що вивчаються в теорії надійності:

Справний стан – коли елемент відповідає усім вимогам нормативно-технічної та конструкторської документації.

Несправний стан – елемент не відповідає всім вимогам (хоча б одній).

Працездатний стан - стан елемента, в якому відповідає вимогам значення всіх характеристик, пов'язаних зі здатністю системи виконувати всі свої задані функції.

Непрацездатний стан – стан елемента, в якому не відповідає вимогам нормативно-технічної та конструкторської документації хоча б одна характеристика, яка показує адекватність системи виконувати визначені функції.

Граничний стан – якщо елемент перебуває в цьому стані, то відновити працездатність такого елемента неможливо, а подальша експлуатація недопустима або неможлива. [2]

Відмова - це певна подія, після виникнення якої система не може зберігати свій працездатний стан.

Безвідмовність - здатність системи безперервно підтримувати працездатний стан протягом часу.

Довговічність – здатність підтримувати працездатність протягом певного терміну при умові проведення вчасного технічного обслуговування.

Ремонтопридатність – здатність елемента чи системи підтримувати та відновлювати свій працездатний стан завдяки технічному обслуговуванню чи ремонту.

Збережність - здатність елемента чи системи зберігати значення параметрів, що характеризують здатність об'єкта виконувати необхідні функції протягом і після зберігання та транспортування.

Надійність є комплексною характеристикою, що, в залежності від призначення об'єкта та умов його застосування, може включати безвідмовність, довговічність, ремонтопридатність і збережність або певні поєднання цих властивостей.[3]

Для об'єктів, які можуть стати джерелом загрози, важливим поняттями є безпека і живучість.

Безпека - властивість об'єкта залишатися безпечним для життя і здоров'я людей при його виготовленні, використанні чи навіть після порушення працездатного стану

Живучість - властивість об'єкта протистояти розвитку та появі критичних відмов при умові дотримання встановлених умов експлуатації, обслуговування і ремонту. Також може визначатися як властивість зберігати обмежений(можливо, деградуючий) працездатний стан при виникненні певних умов експлуатації, що не були визначеними наперед. [4]

## 1.2 Показники надійності

Теорія надійності визначає певні кількісні характеристики та показники. Це необхідно для того, щоб:

- враховувати вплив використанні окремих елементів в системах на надійність таких систем;
- можна було визначити вимоги до надійності систем, елементів чи пристроїв, що проєктуються;
- ефективно порівнювати між собою варіанти побудови системи;
- правильно розраховувати необхідний кількість запасних частин що можуть знадобитися за термін служби, а також сам термін служби пристрою.

Показники надійності - це кількісні характеристики властивостей, з яких в свою чергу складається надійність певного об’єкта.

Розрізняють наступні показники надійності:

- Одиничний показник надійності - це показник надійності, який показує якусь одну з властивостей, з яких складається надійність об'єкта.
- Комплексний показник надійності - що показує декілька властивостей, з яких складається надійність об'єкта.
- Розрахунковий показник надійності - це показник надійності, який визначається розрахунковим методом.
- Експериментальний показник надійності - це показник надійності, значення оцінка якого визначається за даними випробувань.
- Експлуатаційний показник надійності - це показник надійності, значення якого визначається за даними що отримують під час експлуатації.
- Екстрапольований показник надійності - це показник надійності, значення якого визначається на підставі результатів розрахунків і

експлуатаційних даних, шляхом екстраполяції на інші умови роботи. [5, 6]

## 1.3 Етапи розрахунків надійності елементів і систем

Основні види розрахунку надійності показані на рисунку 1.

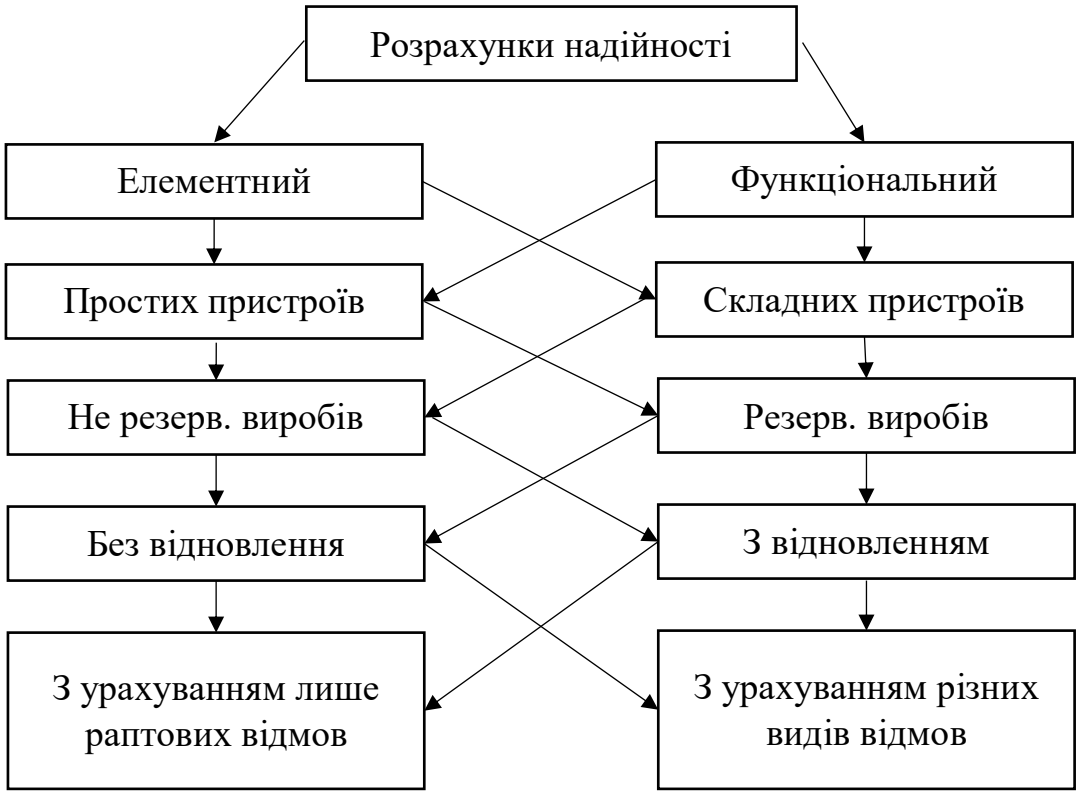


Рисунок 1 – Види розрахунків надійності

Для прогнозування надійності проєктованої системи та виявлення слабких місць пристрою до його виготовлення розрахунки надійності починають проводити уже на етапі проєктування пристрою.

З метою оцінки кількісних показників надійності уже спроєктованої системи та визначення її відповідності заданим вимогам до надійності проводять розрахунки на етапах випробувань і експлуатації. [8]

Існує багато різних методів визначення надійності.

При цьому:

- елементна надійність - це визначення показників надійності виробу, на основі комплектуючих частин, що входять до пристрою;
- функціональна надійність - це визначення показників надійності що стосуються виконання пристроєм спроєктованих функцій.

# 2 ОГЛЯД МЕТОДІВ МОДЕЛЮВАННЯ НАДІЙНОСТІ

Для систем будь-якої складності, використання та створення моделей надійності допомагає більш об’єктивно та точно приймати рішення під час розробки пристрою.

Для розуміння надійності системи потрібно знати:

- надійність окремих елементів цієї системи;
- взаємозв’язки елементів системи в критичних ситуаціях.

Моделі надійності допомагають визначати слабкі ділянки та, в результаті, забезпечувати заданий вимогами рівень надійності.

Деякі відомі моделі:

1. блок-схеми надійності;
2. аналіз дерева відмов;
3. аналіз дерева успіху;
4. моделі Маркова;
5. моделі Петрі;
6. моделі фізики відмови.

## 2.1 Блок-схеми надійності

Блок-схема надійності (БСН) - це графічно-математичне представлення певної системи, що дозволяє оцінити надійність системи в цілому виходячи з характеристик окремих компонентів.

Модель складається з блоків, кожен блок представляє підсистему чи компонент системи, що досліджується. Блоки можуть мати послідовні, паралельні та інші зв’язки. Приклад наведений на рисунку 2.

Значення R – це ймовірність надійності елемента протягом певного часу.

Блок-схеми складних комплексних систем можуть будуватися в декілька рівнів. Модель всієї системи буде оглядовою, показуючи зв’язки великих

компонентів, але спрощуючи підсистеми до якогось компонента верхнього рівня. Далі для кожного такого компонента можна побудувати свою, більш детальну, модель яка буде враховувати особливості підсистеми та зв'язки всередині підсистеми. Такий метод допомагає спрощувати планування комплексних систем, розділяючи складнощі системи на декілька команд чи етапів розробки та використовувати такі схеми на будь-якому етапі розробки системи, починаючи з концепту та закінчуючи плануванням кожного модуля.

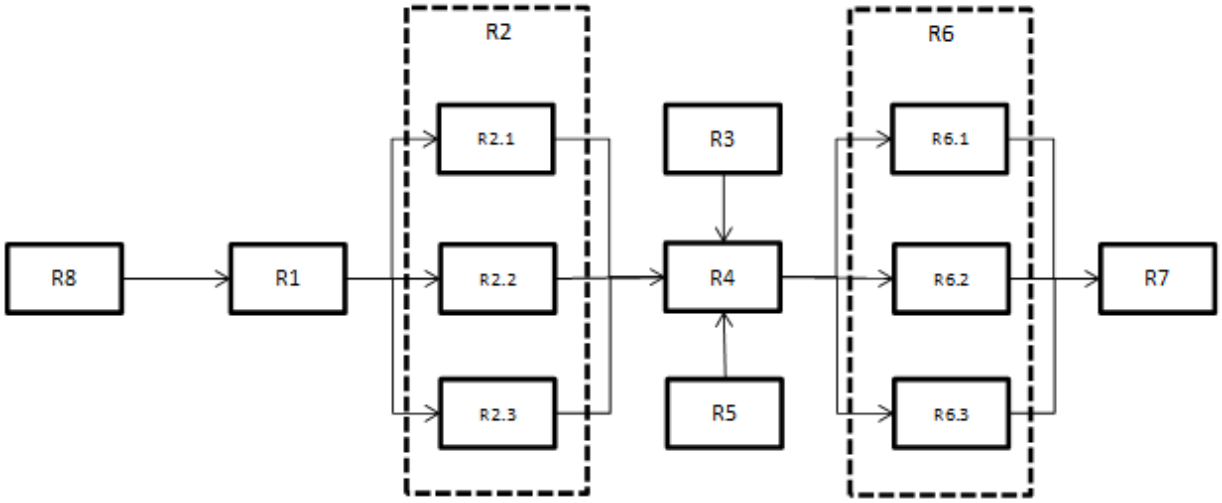


Рисунок 2 – Приклад блок-схеми надійності

Для того, щоб БСН модель була продуктивною, тобто могла реально допомогти в прийнятті рішень, потрібні:

- параметри надійності кожного елементу, такі як ймовірність відмови, тривалість роботи, вимоги до функціональності, середовище роботи;
- попередня оцінка надійності системи, це може як інженерний прогноз, так і дані з проведених тестувань чи документація від виробника;
- БСН повинна оновлюватись синхронно з оновленнями системи, для того щоб вона залишалась релевантною. Кожного розу, як дизайн одного з компонентів оновлюється, модель має отримувати відповідні зміни. Звісно, краще щоб модель переглядалася на постійній основі.

БСН можна використовувати уже на стадії концепції. На ранніх стадіях планування модель важлива для інженерів, щоб команда розуміла варіанти, доступні для забезпечення надійності системи.

Якщо потрібно провести декілька симуляцій для моделювання певних ситуацій, створюються окремі БСН. Для складних систем можуть розроблятися різні моделі для симуляції різних сценаріїв і оцінки впливу навколишнього середовища чи змін в конфігурації на роботу системи.

Також моделі БСН можна використовувати як моделі для ранніх оцінок концепцій для вибору потенційних варіантів досягнення бажаного рівня надійності. БСН модель на ранніх етапах планування може допомогти виявити слабкі ділянки, які потребують якихось змін чи покращень, а також виявляти помилки в дизайні системи.

БСН модель дозволяє команді інженерів фокусуватись на саме тих рішеннях, що мають реальний вплив на надійність розробки. Вона дозволяє знайти слабкі ділянки які потребують покращень, оскільки немає сенсу досягати високого рівня надійності окремих елементів, які насправді не впливають на надійність системи в цілому.

Тому доцільно розглядати БСН моделі саме як інструменти для аналізу компромісів, прийнятих рішень та балансу між надійністю та складністю системи. [9]

## 2.2 Аналіз дерева відмов

Дерево відмов (ДВ) – це графічна модель паралельних та послідовних комбінацій несправностей, які призводять до заданої несправності або помилки. Помилками можуть бути події, спричинені відмовами компонентів, програмними помилками, помилками людини чи будь-які події які призводять до виходу пристрою з ладу. Таким чином ДВ відображує взаємозв'язки між подіями які призводять до несправності.

Аналіз дерева відмов (АДВ) – це аналітична техніка, яка аналізує контекст та середовище роботи системи для пошуку реалістичних ситуацій, коли може відбутися задана небажана подія (небажаний стан системи) в певному критичному стані роботи цієї системи.

Дерево несправностей не є засобом для виявлення помилок чи моделлю усіх можливих критичних ситуацій. Модель дозволяє аналізувати відомі, наперед задані несправності. Одне дерево несправностей пристосоване для конкретної події і показує певний режим відмови системи. Тому модель включає лише ті помилки, які пов’язані з верхньою подією.

Також дерево несправностей  є якісною моделлю, а не кількісною. Кількісні моделі орієнтовані ні виявлення спільних закономірностей, а якісні – на дослідження то пояснення конкретної ситуації.

В основі дерева несправності лежить бінарність – результат є успіхом чи невдачею. Дерево складається з блоків, які називаються воротами. Ворота дозволяють або забороняють проходженню логіки відмов вгору. Таким чином показуються взаємозв’язки між подій, що призводять до верхньої події. Входом воріт є «нижчі» події, відповідно виходом – верхня подія.[10]

Приклад такого дерева наведений на рисунку 3.

Основна перевага АДВ полягає в тому, що аналіз дає унікальне уявлення про функціонування та потенційну несправність системи.

До переваг належать:

1. Допомагає визначити несправності дедуктивним методом, що допомагає команді зосередитись на причинах кожної події в логічній послідовності.

2. Дозволяє виділити важливі елементи системи, пов'язані з відмовою системи. Можна знайти елемент, який призводить до багатьох можливих відмов і покращивши надійність цього елемента значно покращити надійність усієї системи.

3. Візуалізації покращують розуміння системи та допомагають командам розробників в комунікації.

4. У порівнянні до БСН модель можна вважати альтернативним методом системи та надає інструмент, який фокусується на режимах відмов один за одним.

5. Допомагає зосередитеся на одній несправності за раз. Команда вибирає область для фокусування на початку аналізу.

6. Модель дозволяє вивчити багато способів виникнення несправності та може виявити неочевидні шляхи до невдачі, які пропускають інші підходи до аналізу.

7. Дозволяє враховувати людський фактор. [11]

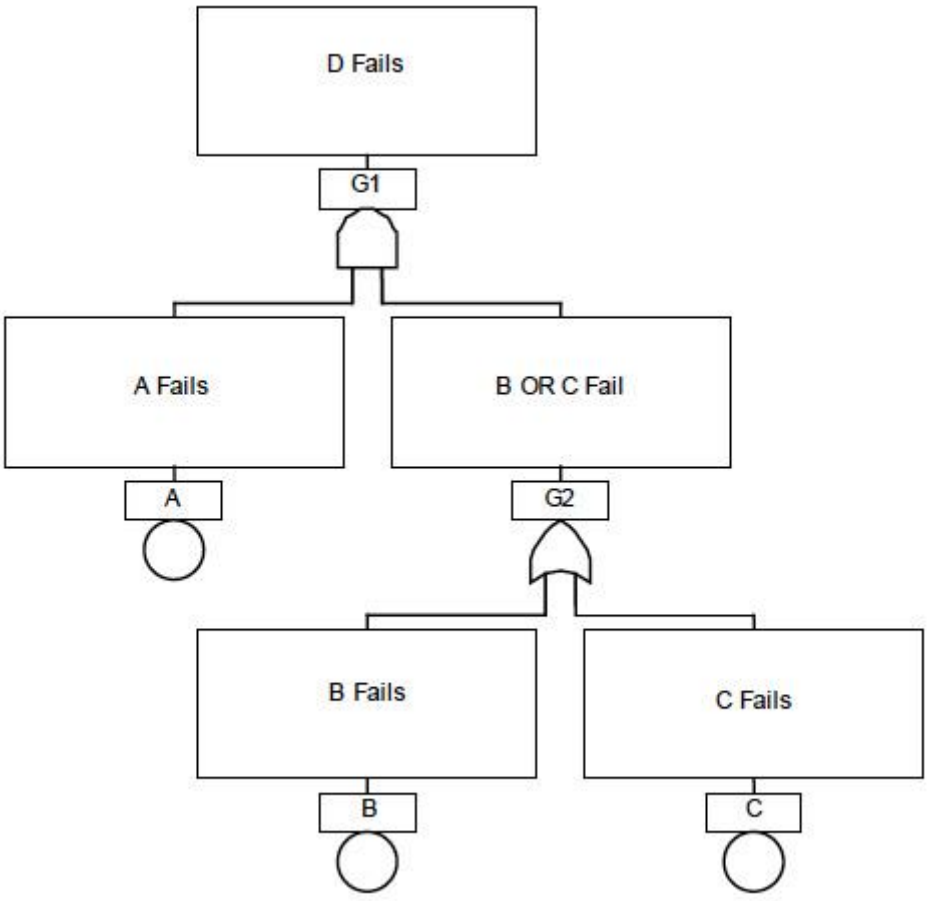


Рисунок 3 – Приклад дерева несправностей

## 2.3 Аналіз дерева успіху

Аналіз дерева успіху дуже схожий до АДВ, з тією лише різницею, що вища подія є успіхом а не несправність.

Замість того, щоб зосереджуватись на тому як модель може вийти з ладу, увага приділяється співвідношенню елементів та реакції на події таким чином, щоб система працювала як очікується.

## 2.4 Моделі Маркова

Моделі Маркова названі так в честь російського математика Андрія Маркова. Спочатку цей термін стосувався тільки математичних моделей, в яких майбутній стан системи залежить тільки від попереднього стану. Властивість системи не залежати від історії переходів називається властивістю Маркова. У більшості реальних пристроїв та електронних компонентів є певні постійні показники відмов, що дозволяє використовувати моделі Маркова для аналізу надійності таких пристроїв.

Такі моделі мають істотну перевагу при аналізу надійності, а саме те, що вони дозволяють враховувати не лише несправність, а й час ремонту. Завдяки цьому модель можна використовувати для оцінки надійності пристроїв із встановленими стратегіями технічного обслуговування. Для відмовостійких систем можливість враховувати ремонт та роботу під час обслуговування, коли система не повністю справна, але все ще функціональна, має дуже велике значення. Такі моделі підходять для вирішення задачі прогнозування інтервалів технічного обслуговування, необхідних для забезпечення поставленого рівня надійності. [12]

Модель Маркова складається з множини всіх можливих станів системи та множини можливих переходів між станами (транзитів). Переходи можуть мати свої параметри, такі як швидкість переходу. Наприклад, швидкість

переходу від непрацюючого до працюючого стану може позначати час ремонту. Таким чином можна побудувати найпростішу модель Маркова – з двома станами та переходами (відмова та ремонт). При графічному зображенні моделі Маркова, стани позначаються кругом, а переходи між станами – стрілками. Приклад простої моделі показаний на рисунку 4.

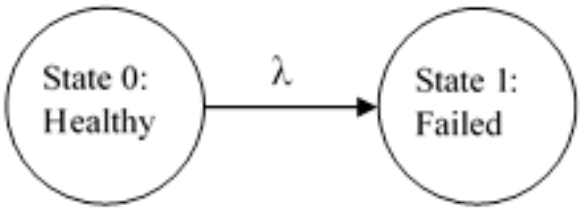


Рисунок 4 – Мінімальна модель Маркова

Швидкість переходу від одного стану до іншого позначається символом λ. Нахай, ймовірність того, шо в певний момент часу t система перебуває в певному стані j, дорівнюватиме $P_j$ (t). Так як спочатку пристрій знаходиться в робочому стані, початкове значення ймовірність перебувати в стані P0 (0) = 1 і початкове значення P1(0) = 0. Ці ймовірності не є сталими і змінюються в часі відповідно зі швидкістю переходу λ. Через певний час dt ймовірність здійснення системою переходу зі стану 0 у стан 1 λdt. Тоді ймовірність переходу від стану 0 в стан 1 протягом певного додаткового інтервалу часу dt, визначається добутком ймовірності перебування в стані 0 на момент початку відліку часу для цього інтервалу на ймовірність переходу протягом інтервалу dt.

$$dP_0 = -(P_0)(\lambda dt)$$

Якщо поділити обидві сторони рівняння на dt, в результаті отримаємо просте диференціальне рівняння

$$\frac{dP_0}{dt} = -\lambda P_0$$

Зазвичай в реальних моделях Маркова є стан, що відповідає повній працездатності системи, коли немає жодних збоїв чи несправностей, та повний непрацездатний стан, коли система уже не може виконувати свої функції. І решту станів можна вважати проміжними, що відповідають за часткові несправності та вихід з ладу окремих компонентів. Переходи між проміжними станами показують шлях системи від повного працюючого стану до повного несправного та зменшують ймовірність системи перебувати в поточному стані і, відповідно, збільшують ймовірність системи перейти в інший стан зі швидкістю, прямо пропорційною параметру λ та ймовірності поточного стану.

Для кожного стану можна записати певний вираз, рівняння стану. Цей вираз характеризуватиме темп зміни ймовірності цього стану, враховуючи ймовірності переходів в цей стан та переходів з цього стану. Всі переходи в певний стан можна позначити як через сумарний потік ймовірності переходу в цей стан. Він обчислюється як сума добутків швидкостей переходу та ймовірностей перебування в цьому стані на початку переходу. Сумарний потік переходу з стану обчислюється як сума добутків всіх переходів зі стану на ймовірність цього даного стану. Приклад зображений на рисунку 5.

Зазвичай на моделях символом λ позначається коефіцієнт поломки, а символом μ – коефіцієнт ремонту. Для стану k можна записати рівняння, що визначає часову похідну $P_k(t)$, що дорівнює сумі потоків в даний стан і з нього. Таким чином

$$\frac{dP_k}{dt} = \sum_i \lambda_{i,k} P_i + \sum_n \mu_{n,k} P_n - \left( \sum_j \mu_{k,j} + \sum_n \lambda_{k,n} \right) P_k$$

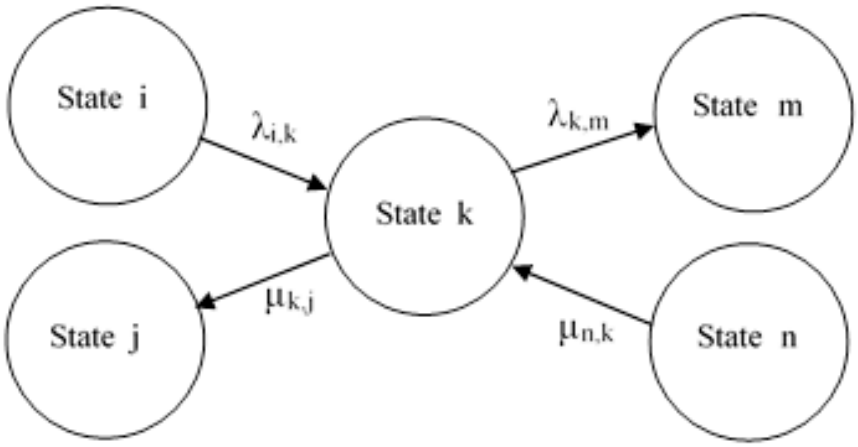


Рисунок 5 – Приклад моделі Маркова

Кожен стан моделі має власне рівняння такої форми. Набором таких рівнянь і визначається поведінка системи. [13]

## 2.5 Мережі Петрі для моделювання надійності системи

Мережі Петрі – один з методів математичного моделювання, що використовують для моделювання поведінки розподілених систем.

Перевагою підходу є те що модель може відобразити як елементи в справному стані, та і несправні. Завдяки цьому модна будувати моделі які відповідатимуть системі з частковою деградацією, коли частина елементів вийшли з ладу, знаходяться на обслуговуванні чи ремонті.

В цьому є перевага методу перед блок-схемами надійності та аналізом дерева відмов. Зазвичай в цих методах припускається, що час ремонту системи є коротким, тоді як мережі Петрі дозволяють моделювати систему що знаходиться на тривалому ремонті і продовжує функціонувати.

Таким чином метод поєднує в собі моделювання функціональності та моделювання надійності системи.

Існує багато видів моделей Петрі, серед них:

– стохастична – переходи мають значення затримки, що обирається випадковим чином;
– часова - переходи мають значення затримки;
– функціональна – переходи мають значення функцій;
– кольорова – мітки можуть бути різних типів, різні типи позначаються різними кольорами;
– ієрархічна – переходи можуть містити інші вкладені мережі, переходи яких, в свою чергу, також можуть мати вкладені мережі.

Приклад такої моделі показаний на рисунку 6.

Смужками позначаються переходи, через які проходять мітки. Спрацювання переходу називається подією.

Колами позначають позиції, в яких можуть знаходитися мітки. Позиції одного типу мають бути розділені переходом.

Стрілками чи дугами позначають потік міток між позиціями та переходами.

Мітки можуть зберігатися в позиціях та можуть рухатися по всьому графу. Мітки не можуть зберігатися в переходах. Події не можуть відбуватися без достатньої кількості міток.

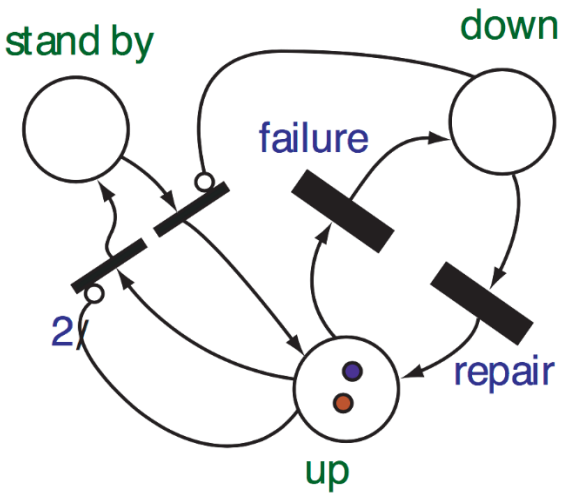


Рисунок 6 – Приклад мережі Петрі

В графі, наведеному на рисунку 6, показана система, що складається з двох одиниць(модулів), що представлені мітками. Кожен модуль може бути в трьох станах, працюючому, несправному, та готовому до роботи. Ці стани показані позиціями(колами). В переходах показані переходи, такі як поломка чи ремонт, їх частота та тривалість. [14]

## 2.6 Вступ до фізики моделей відмов

Моделі фізики відмови (ФВ) – практичні методи проєктування систем, що базуються на розумінні процесів, що спричинюють помилки чи відмови.

В моделях фізики відмов використовуються знання про властивості системи для пошуку рішень підвищення надійності. Ці моделі корисні для:

- прогнозування часу безвідмовної роботи;
- аналізу компромісів;
- вибору кращих матеріалів та технологій;
- планування стратегії пом'якшення наслідків збоїв;
- пришвидшення тестування;

Моделі ФВ використовують комп’ютерні симуляції, дані експериментів та вимірювань та статистичні оцінки для прогнозування впливу середовища та умов використання на ефективність та надійність пристрою.

Можна виділити два основні підходи: емпіричний та детермінований.

В основі детермінованого підходу лежить визначення можливих причин відмов. Потрібно враховувати такі дані, як завантаженість системи, властивості матеріалу, геометричну конструкцію, явища деформації та корозії, можливість деградації, ерозію, дифузію – все це може спричинити раптову поломку.

Також поломки можуть виникати унаслідок накопичення пошкоджень предмета.

Детерміновані моделі намагаються моделювати процеси і переходи, через які проходить система до відмови.

Емпіричні моделі прогнозують час до відмови при певних стресових умовах.

Емпіричний підхід не моделює внутрішні процеси системи, а зазвичай використовується на основі статистичних даних що отримують в польових дослідженнях та тестуваннях. Прикладом емпіричної моделі є співвідношення температури і вологості Пека. [15]

# 3 ГРАФО-ЛОГІЧНА МОДЕЛЬ

Графо-логічна модель (GL-модель) – концепція, що поєднує в собі ідеї використання для моделювання надійності багатопроцесорних систем неорієнтованих графів та логічних функцій, які є ребрами таких графів. Модель встановлює залежність справності чи несправності системи в потоці відмов зі зв'язністю графа. Якщо граф зв'язний, то це означає що система працює зі збереженням своєї функціональності, якщо граф втратив зв'язність – система несправна. Серед переваг даного підходу можна виділити простоту структури моделі, яка досягається завдяки перенесенню частини складності структури на реберні функції.

Модель системи, що стійка до певного числа відмов (k-out-of-n система), називається базовою. Базова система зберігає функціональність якщо вийшли з ладу k або менше елементів.

Кількість елементів базової системи, які можуть вийти з ладу не порушивши її функціональності, називається її кратністю.

## 3.1 Базові GL-моделі

Для прикладу візьмемо випадок, в якому система складається з *n* модулів та зберігає свою функціональність при виході з ладу *k* або менше елементів. Модель буде відповідати k-кратній відмовостійкій мультипроцесорній системі (ВМС), або ж просто k-ВМС.

Для такої k-ВМС побудуємо певний граф G, який э зв’язним та неорієнтованим. Ребра цього графа позначимо набором булевих функцій $f_1(x_1,\ldots,x_n),\ldots, f_L(x_1,\ldots,x_n)$. У відповідність до стану кожного модуля ВМС у відповідність виберемо індикаторну змінну $x_i$ $(i = \overline{1,n})$ за наступним правилом - якщо *i*-ий елемент ВМС несправний, значення відповідного індикатора дорівнюватиме нулю. Якщо елемент справний – одиниці. У графі присутні

лише ті ребра, для яких значення відповідних їм булевих функцій дорівнюють одиниці. Якщо при змінні індикаторної змінної значення функції змінюється – відповідне ребро додається чи видаляється.

Так як для адекватної роботи моделі одним з основних понять є зв'язність графа, булеві функції мають бути такими, що зв'язність графа втрачатиметься якщо і тільки якщо не менше k+1 індикаторних змінних матимуть значення нуля. Функції $f_1, \dots, f_L$ потрібно вибрати так, щоб виконувалося наступне: повинна існувати множина таких реберних функцій з нульовим значенням, що видалення з графу відповідних ребр призводитиме до втрати графом зв'язності, для всіх наборів індикаторних змінних, що містять більше ніж k нульових змінних.

Очевидно, саме вибір правильних реберних функцій є основним критерієм для побудови правильно працюючої GL-моделі.

Однією з найпростіших і рекомендованою для організації GL-моделі структур є граф циклічного типу. Приклад найпростішої GL-моделі можна отримати, якщо взяти n-реберний циклічний граф ребрами якого будуть такі булеві функції, що $f_i = x_i$. В результаті отримаємо базову 1-ВМС (рисунок 7). Саме циклічним графам приділятиметься увага далі.

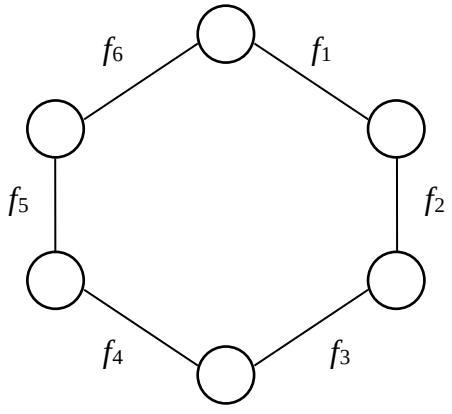


Рисунок 7 – Приклад GL-моделі з циклічним графом

В загальному випадку стан k-ВМС, що складається з n модулів, стан системи визначається як відношення m несправних до n-m справних модулів. У відповідність до кожного стану такої системи можна обрати певну невпорядковану m-вибірку з n-елементів.

Розіб'ємо множину елементів моделі на підмножини, так що $n_u = \{\alpha_1, \alpha_2, \ldots, \alpha_{n_u}\}$. Для k-ВМС визначимо множину що складається з усіх $m_v$ вибірок з $n_u$ елементів та позначимо її як $K(m_v,n_u)$.

Тепер позначимо $K(s, n_p) = \{a_1, a_2, \ldots, a_g\}$, де $a_i$ –S-вибірка з $n_p$ $(i = \overline{1, g})$, а $K(t, n_r) = \{b_1, b_2, \ldots, b_h\}$, де $b_j$ – t-вибірка з $n_r$. Визначимо добуток множин $K(s, n_p)$ і $K(t, n_r)$:

$$K(s, n_p) \times K(t, n_r) = \{(a1 \times b1), (a1 \times b2), \ldots, (a1 \times bh), \ldots, (ag \times b1), (ag \times b2), \ldots (ag \times bh)\}$$

де $(ai \times bj)$ – конкатенація вибірок $a_i$ і $b_j$. Тепер візьмемо n-множину всіх елементів системи та розіб'ємо її на q підмножин, так щоб вони не перетиналися та щоб в першій підмножині було $n_1$, в другій – $n_2$ елементів, в $q$-ій - $n_q$ елементів, тобто для довільного члену $n = \sum_{i=1}^{q} ni$.

$$\sum_{m_1,\ldots,mq} C_{n_1}^{m1} \cdot C_{n2}^{m2} \cdot \ldots \cdot C_{nq}^{m_q} = C_n^m$$

Тоді запишемо рівність

$$K(m, n) = \bigcup_{m_1, m_{2,\ldots,} m_q} K(m_1, n_1) \times K(m_2, n_2) \times \ldots \times K(m_q, n_q)$$

Тепер можна перейти до побудови GL-моделі k-ВМС використовуючи наведене розбиття. Для забезпечення умови несправності, а саме втрати графом зв'язності, критерієм відмови базової k-ВМС можна записати втрату кільцевим графом двох і більше ребр, для чого потрібно забезпечити рівність двох і більше булевих реберних функцій нулю.

Для вибору правильних функцій визначена теорема:

Візьмемо k-ВМС, що складається з n елементів, і множина цих елементів розбита описаним вище способом на q так о кількість елементів в цих підмножинах дорівнюватиме елементів $n_1$, $n_2$ ... $n_q$. Існують $f_1(x_1, \dots, x_n), \dots, f_L(x_1, \dots, x_n)$ – булеві функції, визначені таким чином, що якщо хоча б m змінних, що належать до підмножини яка визначає стан $m_i$ елементів для кожного з $n_i$ підмножин $(i \in \overline{1, q})$ відповідно до значень чисел $m_1, m_2, \dots, m_q$ з і-го набору то відповідна функція $f_j (j \in \overline{1, L})$ прийматиме значення нуля. Також справедливим є таке твердження: якщо хоча б m+1 з n змінних дорівнюють нулю, то не менше двох функцій з множини $f_1, \dots, f_L$ матимуть значення нуля. [16]

Розглянемо план побудови GL-моделі, що описує поведінку k-ВМС та вибір булевих функцій $f1(x_1, \dots, xn), \dots, f_L(x_1, \dots, xn)$ так, щоб вони відповідали умовам приведеної теореми:

1. Розбиваємо множину з $n$ елементів, з якої складається k-ВМС на $q$ множин так, щоб вони не пересікалися та позначимо кількість елементів відповідно $n_1, n_2, \dots, n_q$ так що $(\sum_{i=1}^{q} ni = n)$.
2. Для $m$ потрібно перечислити всі можливі варіанти значень $m_1, m_2, \dots, m_q$ так що $\sum_{i=1}^{q} mi = m$, $m_i \leq n_i$.
3. Для кожного значення m записати вираз типу

$$K(m_1, n_1) \times K(m_2, n_2) \times \dots \times K(m_q, n_q) =$$
$$= \{(a_1 \times b_1 \times \dots \times c_1), (a_1 \times b_1 \times \dots \times c_2), \dots, (an1 \times bn2 \times \dots \times cnq)\},$$

де $a_i, b_{j,\dots,}c_k$ – це $m_1$ – вибірки з $n_1$-множини, $m_2$ – вибірки з $n_2$-множини, …, $m_q$ – вибірки з $n_q$-множини ($i = \overline{1, C_{n_1}^{m_1}}, j = \overline{1, C_{n_2}^{m_2}}, \dots, k = \overline{1, C_{n_q}^{m_q}}$) тобто $a_i$ вибірка вигляду $\{\alpha_1, \alpha_2, \dots, \alpha_{m1}\}$, $b_j$ вибірка вигляду $\{\beta 1, \beta 2, \dots, \beta m2\ \}$, …,$c_k$ вибірка вигляду $\{\gamma_1, \gamma_2, \dots, \gamma_{mq}\}$.

4. Записати булеві вирази на основі виразу з п.3. Булева функція $f_i(x_1, \dots, x_n)$, де $i$ – номер набора значень чисел $m_1$, $m_2$, …,$m_q (i = \overline{1, L})$,записується в вигляді:

$$f_i = (\wedge_i y_{a_i}) \vee (\wedge_j y_{b_j}) \vee \dots \vee (\wedge_k y_{c_k})$$

де $y_{a_i} = \vee_{\sigma=1}^{m_1} x_{\alpha_\sigma}, y_{\beta_j} = \vee_{\theta=1}^{m_2} x_{\beta_\theta}, \dots, y_{c_k} = \vee_{\omega=1}^{m_q} x_{\gamma_\omega}$

Розглянемо такий варіант розбиття n – множини, який спрощуватиме завдання запису функцій $f1, \dots, f_L$. Дійсно, існують такі варіанти, коли для всіх підмножин $K(m_J, n_J)$ справедливий один з наступних випадків:

1. Якщо $m_j = 1$, отримаємо $y_{a_t} = x_{a_t}, t = \overline{1, n_j}$. Тобто $f_{mj}$ можна записати як елементарні кон'юнкції:

$$f_{m_j} = x_{\alpha_1} \wedge x_{\alpha_2} \wedge \dots \wedge x_{\alpha_{nj}}.$$

2. Якщо ж $m_j = n_j$, отримаємо $y_{a_t} = x_{a_t}, t = \overline{1, n_j}$. Тобто $f_{mj}$ можна записати як елементарні диз'юнкції:

$$f_{m_j} = x_{\alpha_1} \vee x_{\alpha_2} \vee \dots \vee x_{\alpha_{nj}}.$$

Для отримання однієї з перерахованих форм розбиття розбиття n-множин можна проводити поетапно. Спочатку в результаті першого розбиття матимемо $n_1, n_2, \dots, n_q$ підмножин. Після цього відбираємо ті підмножин $n_i$, вираз яких не

підходить ні під $K(1, n_j)$, ні під $K(n_j, n_j)$ і виконуємо для низ повторне розбиття і так далі.

### 3.2 Мінімізація

Для втрати кільцевим графом зв'язності достатньо щоб він втрачав два ребра, тобто дві реберні функції відповідали значенню нуль. Проте базова модель побудована за описаним планом частіше втрачатиме більше двох ребр, тому актуальним є питання мінімізації отриманої моделі.

Для мінімізації такої моделі доцільно провести склеювання, подібно до тієї, що використовується в методі Квайна:

$$\left.\begin{array}{l} F_1 = A \vee \varphi_1 \\ F_2 = A \vee \varphi_2 \end{array}\right\} \Rightarrow F_3 = A \vee \varphi_1 \cdot \varphi_2$$

Тут A, $\varphi_1\varphi_2$ – булеві вирази, що входять до складу реберних функцій F1, F2, F3 GL-моделі. Для проведення таких склеювань необхідною умовою є присутність спільної частини, однакового виразу A відділеного операцією диз'юнкції від решти функції в кожній з функцій які намагаються склеїти. Після цього ці дві функції (F1 та F2) можна замінити на результат склейки, зменшивши кількість ребр в моделі на 1.

Вирази $A$, $\phi_i$ – ДНФ, так як реберні функції цієї моделі представляються у вигляді диз'юнктивної нормальної форми.

У випадках, в яких функції F1 і(або) F2 брали участь в попередніх склеюваннях, це відношення все одно залишається справедливим.

Варто зазначити, що завдяки методиці побудови канонічних GL-моделей, реберними функціями є безповторні ДНФ.

Завдяки цьому можна стверджувати, що множини змінних, які беруть участь в виразах, які знаходяться по різні сторони від знаку диз'юнкції функції F3, не перетинаються.

Важливо відмітити, що мінімізація склеюванням не впливає на здатність k-ВМС втрачати зв'язність при нульових значеннях k+1 індикаторних змінних моделі.

При появі будь-якого вектору стану з (k + 1) нулями, на якому дві функції, що беруть участь в склеюванні, отримують нульове значення, то стає рівною нулю ще хоча б одна функція GL-моделі, при чому така, яка не братиме участі в майбутніх склеюваннях з цими двома функціями.

### 3.3 Додаткові ребра

Використання моделі не обмежується базовими моделями. Зазвичай конкретна ВМС може мати більшу чи меншу відмовостійкість, залежно від певних векторів помилок. ВМС може бути більш стійкою до відмови певного набору модулів реконфігуруватися при появі таких помилок для збереження функціональності. Наприклад, система відмовостійкість системи кратна трьом в більшості випадків. Проте до відмови певного модуля вона більш стійка і на векторах помилок що включають цей модуль, вона поводиться себе як система 4-ВМС, при чому це не обов'язково стосується усіх векторів з чотирма нулями що включають цей модуль. І навпаки, для відображення критично важливого модуля потрібно щоб система поводилася як 1-ВМС чи 2-ВМС при його відмові. Ці умови можуть диктуватися характеристиками модулів самих модулів та їх здатністю виконувати функції інших модулів що відмовляють, архітектурою чи конфігурацією системи та іншими факторами. Очевидно, для моделювання таких систем базової моделі недостатньо. Так як, по суті, GL-моделі складаються всього з двох компонентів – вершин (графа) та реберних

функцій, доступні два варіанти розширення можливостей моделі – ускладнення структури графа та ускладнення логічних функцій. Розглянемо перший варіант.

Нехай задана певна сукупність векторів для яких потрібно змінити певну базову GL-модель. Якщо змінювати модель крок за кроком так, що поведінка моделі змінюватиметься тільки для конкретного вектору, то задачу можна спростити до побудови базової моделі, що зберігає працездатність на конкретному векторі помилок.

Вирішити цю задачу можна додаючи до моделі додаткові внутрішні ребра.

Додаткове внутрішнє ребро r з функцією f, що з'єднує дві будь-які вершини GL-моделі, збільшує ступінь відмовостійкості моделі на певному наборі векторів індикаторних змінних.

Для того щоб функція не впливала на поведінку моделі на інших наборах векторів достатньо щоби вона приймала значення нуля на таких наборах, так як тоді вона видаляється з графа. Іншими словами, потрібно забезпечити появу такого ребра тільки для заданих наборів векторів індикаторних змінних, додаючи нові шляхи на графі.

Далі:

1. Нехай з'являється вектор стану вектор α і при цьому з графа випадають ребра $a_i$ та $a_j$ . Тоді додаткове ребро r для забезпечення зв'язності повинно проходити між ними. В іншому випадку додаткове ребро не забезпечуватиме додаткового шляху до вершин що випадають.
2. В ситуації коли при появі вектору стану α випадають декілька ребр $a_1$, $a_2$, … $a_k$, то ребра $r_1$, $r_2$, … $r_L$, що додаються до моделі, мають додаватися таким чином, щоб не знаходилося більше одного ребра $a_i \in \{a_1, a_2 \dots a_k\}$ між двома найближчими вершинами, інцидентним двом довільним додатковим ребрам.
3. Для забезпечення зв'язності графа при появі вектору α ребру r (чи ребрам $r_1$, $r_2 \dots r_L$) можна функцію

$$Q = \bar{x}_1 \cdot \bar{x}_2 \cdot \ldots \cdot \bar{x}_{m+1} x_{m+2} \cdot \ldots \cdot x_n$$

Тобто конституенту одиниці. [17]

Так як конституента одиниці приймає ненульове значення лише на одному наборі, таке ребро не змінюватиме поведінку моделі на всіх інших векторах, оскільки на таких наборах додаткове ребро буде відсутнім. А для набору для якого конституента записана, ребро забезпечить додатковий шлях між ділянками графа, які втрачають зв'язність через зникнення ребр.

Проведемо в базовій моделі для вектору α внутрішнє ребро $r_1$ з певним логічним виразом. Дано певний інший вектор стану β, для якого також потрібно забезпечити відмовостійкість моделі. Якщо певні два ребра $a_i$, $a_j$, які видаляються з моделі при появі вектору β, знаходяться з різних сторін відносно ребра $r_1$, тоді для забезпечення відмовостійкості моделі при появі такого вектору потрібно приписати ребру $r_1$ функцію $f = Q_1 \vee Q_2$. Коли ж ребра $a_i$, $a_j$ знаходяться з однакової сторони відносно ребра $r_1$, то для забезпечення відмовостійкості потрібно провести інше ребро $r_2$.

Тобто, до кожного внутрішнього ребра GL-моделі приводиться у відповідність диз'юнкції конституент векторів станів модулів моделі, для яких це ребро має збільшувати зв'язність моделі.

Отримана функція ребра r в вигляді ДНФ може бути мінімізована відомими методами мінімізації. GL-модель при цьому зберігає свої властивості і характеристики незмінними. При мінімізації можна взяти до уваги той факт, що функція f фактично є частково визначеною. Для векторів станів, при яких ребра що зникають з GL-моделі знаходяться з однієї сторони відносно ребра r, значення відповідної реберної функції може бути вибране довільним чином, так як ні поява, ні зникнення такого додаткового ребра не вплине на функціонування системи.

Розглянемо на прикладі 1-ВМС (як реберні функції виберемо $f_i = x_i$) . Що складається з 4 модулів. Забезпечимо працездатність на векторах відмов { 1,3 } і { 2,3 }.

Для цього достатньо одного ребра з функцією f (рисунок 8).

$$f = \bar{x}_1\bar{x}_3x_2x_4 \vee \bar{x}_2\bar{x}_3x_1x_4$$

Враховуючи також, що функція f не визначена на всіх наборах, пов'язаних зі значеннями $\bar{x}_1\bar{x}_2$ та $\bar{x}_3\bar{x}_4$, отримуємо:

$$f = \bar{x}_3 \cdot (\bar{x}_1 \vee \bar{x}_2)$$

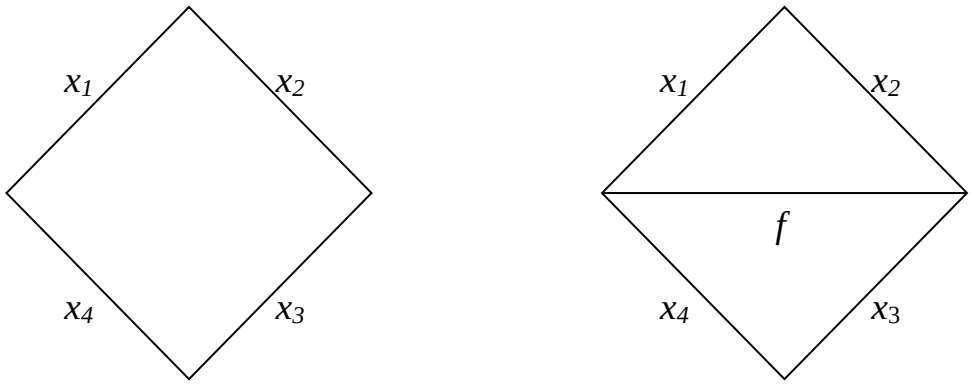


Рисунок 8 – GL-модель з внутрішнім ребром

Також можна відмітити, що функція f не визначена на наборах 1110, 1101, 1011, 0111, тобто на векторах станів системи, коли з ладу виходить не більше одного модуля. Це дозволяє спростити функцію до вигляду $f = \bar{x}_3$.

Даний підхід дозволяє розв'язати задачу побудування GL-моделей для випадків, коли стійкість до певних відмов системи описується довільною множиною векторів станів системи, на яких її працездатність зберігається.

Якщо ж стоїть задача зменшення степені відмовостійкості GL-моделі на певних наборах векторів помилок, одним з варіантів вирішення проблеми є додавання до вершини додаткової вершини та додаткового ребра так, щоб

реберна функція, що з’єднує додаткове ребро з графом моделі, дорівнювала нулю на заданих наборах. Як варіант, такому ребру можна приписати кон’юнкцію конституент нуля для перерахованих наборів.

# 4 ОПИС РЕАЛІЗАЦІЇ ДОДАТКУ

## 4.1 Вибір фреймворку для створення десктоп-застосунків

### 4.1.1 Qt

Qt – безкоштовний крос-платформний фреймворк для створення десктоп-застосунків. Qt був створений 1990 року Гаавардом Нордом та Айріком Чемб-Інгом. Спочатку це був набір класів для мови C++, проте зараз Qt - це фреймворк, який включає у собі:

1. Велику бібліотеку класів для роботи з графічним інтерфейсом користувача, мережею та мультимедіа:
   - Qt core – класи не пов’язані з графікою;
   - Qt GUI – базові класи для створення графічного інтерфейсу, включаючи OpenGL;
   - Qt Multimedia – класи для роботи з відео та аудіо, API для роботи з камерою;
   - Qt Network – API для роботи з TCP/IP;
   - Qt SQL – класи для роботи з SQL базами даних;
   - Qt Test – класи для написання тестів для застосунків побудованих за допомогою Qt;
   - Qt widgets – набір графічних компонентів (віджетів).
2. Qt Creator – власна IDE для створення застосунків. Включає:
   - редактор з підсвіткою синтаксису, перевіркою помилок та системою автозавершення коду;
   - Qmake – автоматизована система збірки програми;
   - Qt Designer – візуальний редактор графічного інтерфейсу, дозволяє будувати стандартизовані графічні додатки за допомогою Qt віджетів;
   - Qt Linguist – система для інтеграції різних локалізацій (перекладів);
   - Qt Assistant – швидкий доступ до офіційної документації.

3. QML (Qt markup language) – починаючи з версії 4.7 стала доступна власна мова для опису графічних компонентів. Мова підтримує як імперативний, так і декларативний стилі. Графічні компоненти описуються в стилі JSON, таким чином дизайн легко імпортувати, наприклад, з інструменту Photoshop. [18]

Основною мовою для розробки довгий час була C++, тепер також доступна мова JavaScript. Також існують окремі проєкти адаптації фреймворку на інші мови програмування:

- Ada (QtAda);
- C# (Qyoto/Kimono);
- Java (Qt Jambi);
- Node.js;
- Pascal;
- Perl;
- PHP (PHP-Qt);
- Ruby (QtRuby);
- Python (PyQt,PySide).

Qt працює на багатьох платформах, офіційно підтримуються наступні:

- Mac OS X;
- Microsoft Windows;
- X Window System (Unix / Linux);
- Вбудовані платформи (PDA, смартфони);
- Windows CE;
- Maemo;
- Symbian.

Фреймворк розповсюджується за двома ліцензіями – безкоштовною та комерційною. За умовами безкоштовної ліцензії, якщо користувач модифікує

код фреймворку, він зобов’язаний надіслати ці зміни розробникам Qt. В іншому випадку можуть виникнути серйозні проблеми з законом, тому якщо користувач не хоче розповсюджувати свої зміни, він може обрати комерційну ліцензію.

За умовами комерційної ліцензії, користувач отримує повні права на вихідний код фреймворку, може модифікувати його за своїм бажанням та зберігати будь-які зміни приватними.

Серед програмного забезпечення, яка використовує Qt, можна виділити:

- Adobe Photoshop Elements;
- Auodesk Maya;
- Dolphin;
- Google Earth;
- Krita;
- VirtualBox;
- VLC Media Player;
- Telegram Desktop.

### 4.1.2 Swing

Swing – Java фреймворк, що надає набір класів та компонентів для створення GUI. Swing входить в бібліотеку базових класів Java (JFC, Java Foundation Classes). Фреймворк було створено як заміну для Java AWT (Abstract Widget Toolkit, перша GUI бібліотека для Java).

Основні можливості Swing:

- незалежність від платформи;
- гнучке налаштування;
- можливість розширення;
- легкий.

Look and Feel – “Look” відповідає за вигляд і “Feel” відповідає за поведінку компонентів, доступні наступні опції:

- крос-платформний – виглядає однаково на всіх платформах, також відомий як стиль “Metal”, обирається за замовчуванням;
- системний – використовується стиль, який є звичним для платформи на якій працює застосунок, тобто стилі будуть різними для Windows, Mac OS та Linux;
- синтетичний – основа для створення власного стилю;
- мультиплекс – можливість комбінувати наведені вище опції. [19]

### 4.1.3 NW.js

NW.js – фреймворк для створення десктоп-застосунків використовуючи веб технології, такі як HTML, JavaScript та CSS.

Фреймворки, побудовані на основі Node.js та Chromium V8 стали дуже популярними, оскільки нова версія JavaScript-інтерпретатора отримала функцію JIT-компіляції та хорошу оптимізацію. Завдяки цьому застосунки, що будуються на основі таких фреймворків, наблизилися за швидкодією до застосунків написаних на C++.

Окрім того, JavaScript є однією з найпопулярніших мов програмування як швидко розвивається. Усі бібліотеки та фреймворки, які широко використовуються в WEB-розробці можуть використовуватися для створення графічних інтерфейсів в NW.js, наприклад:

- React;
- Angular;
- Vue.

NW.js надає доступ до Node.js API та Chrome API.

Серед переваг NW.js:

- Хороша підтримка популярних операційних систем;

- Дозволяє захищати вихідний код (велика проблема усіх JavaScript – проєктів);
- Підтримується компанією Intel;
- Оновлюється через 24 години після виходу нової версії Chromium чи Node.js;
- Стиснення програми до 20 мегабайт (для Windows).

Деякі продукти, які використовують NW.js:

- Qilin – текстовий редактор;
- Gitter – чат для проєктів з відкритим кодом що має пряму інтеграцію з GitHub;
- Construct – популярний інструментарій для створення ігор;
- WeChat Mini Program SDK – популярна в Китаї платформа для створення міні-програм;
- Scout-App – застосунок для перетворення Sass/SCSS в CSS. [20]

### 4.1.4 Electron JS

Electron JS – це фреймворк для створення крос-платформних десктоп застосунків на основі JavaScript, HTML та CSS, розроблений розробником з GitHub, Ченг Жао. Доступними платформами є Linux, Windows та Mac OS X.

Як і NW.js, Electron дозволяє використовувати усі доступні модулі npm. Окрім того, фреймворк надає можливості автоматичних оновлень та звіти про збої.

Серед переваг Electron JS:

- Один з найпопулярніших пакетів для npm;
- Велика екосистема;
- Хороша документація;
- Використовують багато відомих компаній (наприклад, Microsoft);
- Підтримується GitHub;

– Зручна інтеграція з Gulp, Webpack.

Деякі продукти, які використовують Electron JS:

– Discord;
– VSCode;
– Slack;
– Skype;
– Atom;
– WhatsApp;
– Tusk. [21]

Для даної роботи обрано Electron JS, як один з найпопулярніших та найшвидший в розробці фреймворк.

## 4.2 Огляд архітектури

Програма повинна забезпечити наступний функціонал:

1. Формування базових моделей типу K out of N;
2. Мінімізація базової моделі;
3. Можливість вводу векторів відмов;
4. Відображення реакції моделі на введений вектор відмов;
5. Можливість додавання внутрішніх ребр для підвищення відмовостійкості моделі.

Для зручності модель поділено на три частини – саму графологічну модель яка показує зв’язки між вершинами, ребра та їх булеві функції, вектор помилок, значення якого користувач можу змінювати, та граф, який генерується на основі GL-моделі та вектору помилок в даний момент часу та за допомогою якого обчислюється зв’язність GL-моделі при конкретному векторі помилок.

GL-модель що відповідає базовій системі K(m, n), будується на основі введених користувачем чисел m та n, після чого мінімізується методом склеювань, форма для введення вектору помилок міститиме n змінних.

Після цього користувач може вручну додавати набір векторів, до яких GL-модель буде стійкою. У відповідь на це до моделі будуть додані внутрішні ребра, яка відобразяться на моделі та будуть враховані при генерації графа та обчисленні зв'язності моделі.

В додатках представлена схема сутностей та зв'язків у програмі.

K(m, n) – певна вибірка або комбінація вибірок, на основі якої генерується таблиця булевих функцій моделі. Серед методів цієї сутності основними є:

- finite – вирішує, чи вибірка належить одній з трьох кінцевих форм - K(0,n), K(1, n) та K(n, n). якщо вибірка не належить жодній з цих форм, вона розкладається способом, описаним в розділі 3 до множини кінцевих форм;
- genTable() – спочатку за допомогою рекурсії розкладає вибірку до кінцевої форми, і но основі отриманої таблиці генерує булеві функції. Блок-схема алгоритму генерації таблиці наведена в додатках.

Оскільки для правильної роботи моделі потрібно динамічно генерувати функції, основні операції представлені класами мови JavaScript, інтерфейсом яких є три методи:

1. resolve(store) – обчислює значення функції при поточному стані вектору помилок store.
2. toString() – генерує текстове представлення функції для виводу на екран.
3. type() – повертає текстовий код інструкції (D - диз'юнкція, C – кон'юнкція, V - змінна)

Реалізовані наступні примітиви, що відповідають заданому інтерфейсу:

1. Dis – клас, що реалізує диз'юнкцію. Містить вектор булевих функцій (диз'юнкцій, кон'юнкцій чи змінних) *elems*, відповідно метод resolve обчислює диз'юнкцію значень цих функцій.
2. Con – клас, що реалізує кон'юнкцію. Містить вектор булевих функцій (диз'юнкцій, кон'юнкцій чи змінних) *elems*, відповідно метод resolve обчислює кон'юнкцію значень цих функцій.
3. Var – клас, що представляє змінну. Містить індикатор що вказує, чи повинна змінна бути інвертована. Метод resolve обчислює значення змінної відповідно до значення цієї змінної в векторі помилок store.

Така організація дозволяє генерувати функції довільної складності. Також булеві функції не мають прямого доступу до відповідних значень іксів, вони їх отримують лише в момент обрахунку. Завдяки цьому одні і ті ж функції можна використовувати з різними наборами векторів, що буде використано в мінімізації, а дозволивши користувачу змінювати значення цього вектору та перераховувати після цього значення реберних функцій, ми отримаємо динамічну модель.

Отримавши реберні функції, можна будувати базову GL-модель k-BMC. Проте часто кількість реберних функцій модна зменшити, а іноді доволі суттєво. Тому Наступна частина програми – Minimizer з методом minimize, який реалізує алгоритм склейки булевих функцій. Блок схема алгоритму наведена в додатках.

На основі таблиці мінімізованих функцій будується GL-модель.

Спочатку створюються вершини моделі, яким відповідає структура Node. За допомогою метода link вершини об'єднуються в кільцевий граф і до них приписуються згенеровані реберні функції.

GraphLogic - структура, що об'єднує в собі згенеровану структуру вершин та реберних функцій, а також компонент для зберігання та введення вектору помилок XValues (значення «іксів»), на основі яких генерується граф, зо представлений структурою Graph.

Ці структури дозволяють генерувати базову GL-модель k-BMC.

Тепер опишемо процес генерації  та мінімізації внутрішніх ребер, що дозволяють змінювати поведінку моделі на певних векторах помилок.

Блок схема алгоритму додавання внутрішніх в додатках.

## 4.3 Опис реалізації

### 4.3.1 Генерація булевих функцій

Для формування функцій формується структура K(m, n). Для генерації функцій зручно працювати з наступними структурами:

- K(1, n) – кон’юнкція змінних;
- K(n, n) – диз’юнкція змінних;
- K(0, n) – 0.

Тому структура K(m, n) розкладається в таблицю $K(m_1, n_1)$, $K(m_2, n_2)$,…$K(m_i, n_i)$ які належать одному з перерахованих типів.

Розділимо число n на m  частин, так що $n_i = [n / m]$. Так як в такому разі сума всіх $n_i$ може бути меншою за n, збільшимо значення перших n mod m елементів на одиницю. Наприклад:

1. $m = 3, n = 9, n_1 = 3, n_2 = 3, n_3 = 3, 9 \bmod 3 = 0$.
2. $m = 3, n = 8, n_1 = 2, n_2 = 2, n_3 = 2, 8 \bmod 3 = 2$ – збільшуємо $n_1$ та $n_2$ на одиницю. Отримаємо: $n_1 = 3, n_2 = 3, n_3 = 2$.

Для значень $m_i$ потрібно перерахувати всі можливі набори, так що сума значень $m_i$ дорівнює m, і $m_i \leq n_i$. Цю задачу можна умовно розбити на три прості задачі:

1. Розбиття числа m на m доданків, враховуючи варіанти з нулями.
2. Знаходження всіх унікальних перестановок елементів списку.
3. Видалення зі списку перестановок в яких $m_i > n_i$.

В результаті отримаємо таблицю вигляду

$$[K(m_{11}, n_1), K(m_{12}, n_2),\ldots, K(m_{1i}, n_i)]$$
$$[K(m_{21}, n_1), K(m_{22}, n_2),\ldots, K(m_{2i}, n_i)]$$
$$\ldots$$
$$[K(m_{j1}, n_1), K(m_{j2}, n_2),\ldots, K(m_{ji}, n_i)]$$

Очевидно, що не всі структури K матимуть один з трьох бажаних виглядів. Тому процедура розбиття проводиться ще раз для таких елементів. В результаті отримаємо певну вкладену структуру таблиці, з якою працювати не дуже зручно. Зручніше буде обчислити декартовий добуток вкладеної таблиці та рядка, який її містить, збільшивши таким чином кількість рядків в початковій таблиці, але зберігаючи однорівневу структуру.

Цю процедуру повторювати доти, доки усі елементи не матимуть один з перерахованих типів.

Отримана таблиця разом з визначеними типами, які реалізують примітивні булеві функції (Dis, Con, Var), спрощує задачу генерації булевих функцій до наступної:

1. Вибрати рядок таблиці. Для кожного елемента цього рядка, згенерувати одну з двох логічних функцій.
2. Якщо m = 0 – пропустити елемент.
3. Згенерувати диз'юнкцію утворених функцій.

Кожен рядок таблиці представляє окрему функцію.

4.3.2 Склейка булевих функцій

З метою зменшення кількості реберних функцій (якщо можливо), реалізований алгоритм склеювання функцій методом Квайна.

Слід зазначити, що алгоритм пристосований логічних функцій, які є диз'юнкціями функцій (або змінних) без повторів. Згенеровані описаним вище методом функції є саме такими і не втрачають цих властивостей після склеювання.

Склеювання реалізовано за допомогою циклу, який повторюється доти, доки на попередній ітерації буде успішно виконуватися склейка двох функцій.

При склейці функції $f_1$ та $f_2$ видаляються з таблиці та записується функція $f_3$. Тобто кожна успішна склейка зменшує розмір таблиці на одиницю.

Також склеювання виконуються за пріоритетом – обираються функції, які мають «більше спільного» з іншими функціями. Це не впливає на кінцевий результат, але дозволяє спростити реалізацію, а саме виокремлення спільної частини для двох функцій.

1. Обрати функцію f зі списку.
2. Обрати іншу функцію g.
3. Якщо ці дві функції можна склеїти, тобто множина спільних диз'юнкцій не пуста, обрахувати кількість можливих склейок функції g іншими функціями списку.
4. Повторити пункт 3 для всіх можливих функцій g.
5. Якщо множина функцій g не пуста, обрати серед них найкращу та провести склейку. Видалити f та g зі списку. Записати в список отриману функцію. Перейти до пункту 1.
6. Якщо множина функцій g пуста, обрати наступну функцію f та перейти до пункту 2.
7. Якщо не вдалося виконати жодної склейки, алгоритм завершений.

Такі склейки не завжди можливі, наприклад, K(2, 8) генерує 7 функцій які не можна мінімізувати. Проте для K(4, 8) отримуємо 19 функцій, які таким методом можна мінімізувати до 5.

### 4.3.3 Визначення зв'язності графа

Граф генерується на основі стану вектору помилок та функцій на ребрах моделі. На граф переносяться лише ті ребра, які для заданого вектору повертають значення true.

Зв’язність обчислюється емпіричним способом – якщо для всіх можливих пар вершин існує шлях між ними, граф зв’язний.

4.3.4 Проведення внутрішніх ребр

Внутрішні ребра проводяться в моделі для забезпечення додаткової стійкості до обраних векторів помилок.

Першим кроком є перевірка чи потрібно при такому векторі помилок проводити додаткове ребро. Граф ділиться на зв’язні групи, тобто групи вершин, де для кожної пари вершин в такій групі існує шлях між ними. Алгоритм дуже простий:

1. Виберемо довільну вершину x. Це буде перша група.
2. Виберемо наступну вершину y. Якщо існує шлях між x та y, додаємо вершину y до групи.
3. Перевіривши усі вершини, сформуємо першу групу.
4. Виберемо довільну вершину з множини вершин, що не належать першій групі.
5. Аналогічно формуємо другу групу і так далі.

Очевидно, що якщо кількість таких груп менше двох – граф зв’язний і внутрішні ребра проводити немає необхідності.

Далі виберемо дві групи, які можна з’єднати. Умовою для цього є наявність в цих групах двох несусідніх вершин. Тобто двох вершин, між якими нема ребра кільцевого графа моделі (або ж зовнішнього ребра).

Якщо таких вершин немає, то вибирається наступна пара груп.

Якщо між цими вершинами в моделі немає ребр, з’єднаємо їх ребром та припишемо йому конституенту одиниці.

Якщо між цими вершинами вже існує ребро, нова функцій для ребра генерується за допомогою діаграми Вейча.

4.3.5 Генерація функцій для внутрішніх ребер

Спочатку генерується таблиця істинності. Для заповнення таблиці значеннями потрібно лише для кожного вектору помилок для якого генерується функція знайти відповідний рядок в таблиці та записати туди одиницю. За це відповідає клас TruthTable.

Далі на основі таблиці генерується об’єкт класу KMap. Список змінних ділиться на дві частини і для кожного списку змінних генерується послідовність кодів Грея. Наявність такого коду дозволяє створити зручну індексацію колонок діаграми.

В коді Грея змінюється лише один біт. Алгоритм генерації такого коду представлений на рисунку 9. [23]

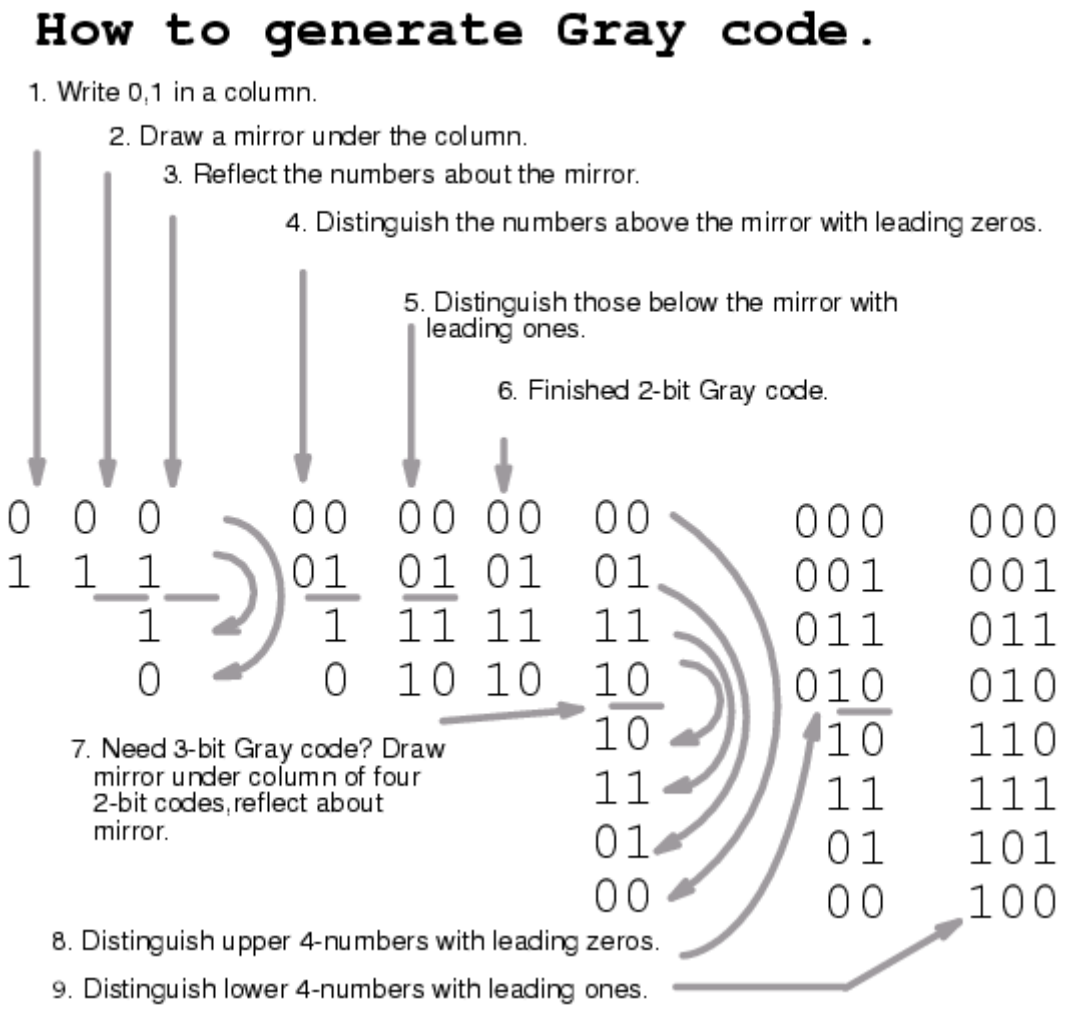


Рисунок 9 – Алгоритм генерації коду Грея

Далі клітинки розбиваються на групи. Для того щоб забезпечити входження кожної клітинки в групу максимального розміру з можливих, виконується наступне:

1. Діаграма покривається певними шаблонами, починаючи з найбільшого (вся таблиця), закінчуючи найменшим можливим (одна клітинка). Шаблони – це можливі групи клітинок.
2. Якщо хоч одна клітинка, яка проходить перевірку шаблоном, має значення нуля, вважається що шаблон не підходить.
3. Шаблон послідовно рухається по таблиці, дійшовши до кінця і перевіривши всі можливі клітинки, шаблон змінюється на інший шаблон такого ж розміру або шаблон меншого розміру.
4. Якщо група клітинок підходить під шаблон, і хоч в однієї клітинки з цієї групи ще немає групи (відслідковується за допомогою прапорця), додаємо шаблон в список та помічаємо клітинки (ставимо прапорець)
5. Якщо всі клітинки з групи уже мають мітку, шаблон група ігнорується, так як уже існує група більшого розміру до якої вони входять.
6. Алгоритм зупиняється після шаблону мінімального розміру (одна клітинка).

Для генерації функції на основі розміченої діаграми для кожної групи зі списку, виділяємо коди Грея для колонок та для рядків, та проводимо їх склейку. Якщо якась змінна міняє значення, вона помічається пропуском (“_”). Наприклад, маючи коди 001 та 011 в результаті склейки отримаємо 0_1. На основі склеєних кодів генеруємо кон’юнкцію:

- 1 – змінна;
- 0 – інверсія змінної;
- _ - пропуск змінної.

В кінці згенеровані кон’юнкції об’єднуються в диз’юнкцію.

## 4.4 Опис інтерфейсу додатку

Розроблений додаток має простий графічний інтерфейс, який дозволяє будувати довільні базові моделі k-out-of-n, модифікувати вектор відмов з негайним відображенням реакції моделі та додавати реберні функції для вибраних векторів відмов з наглядною реконфігурацією моделі. Інтерфейс програми показаний на рисунку 10.

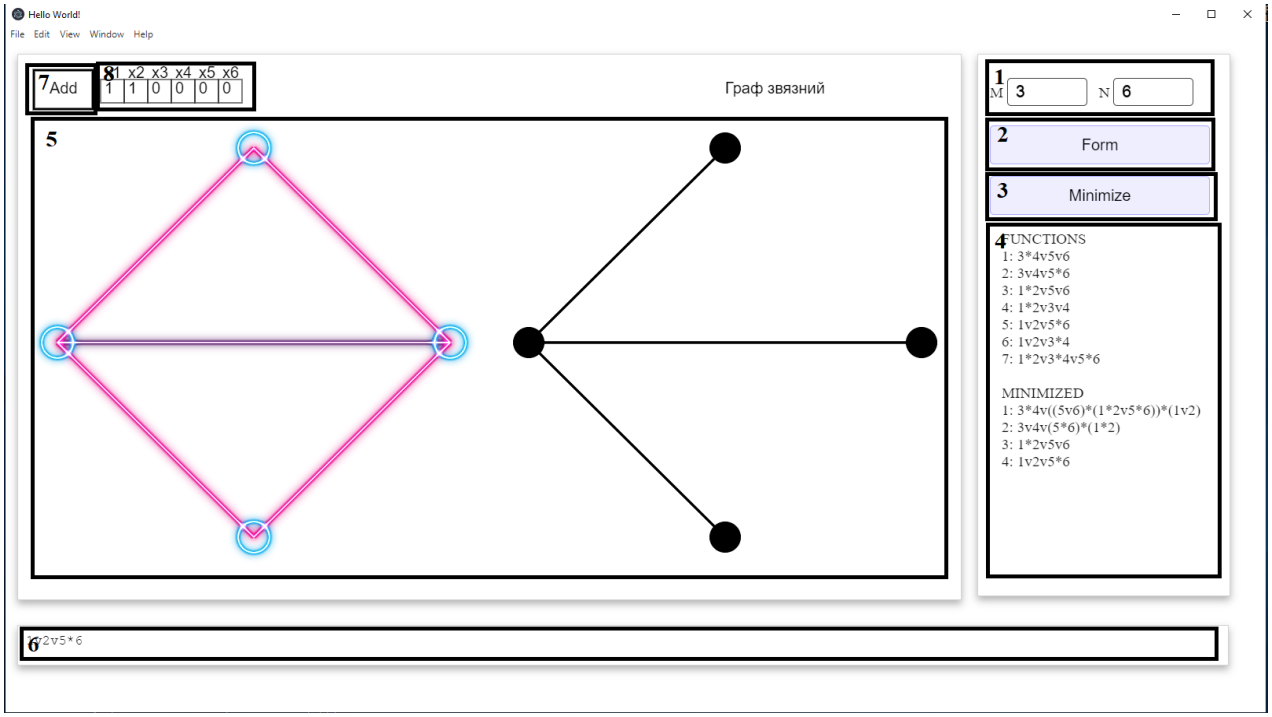


Рисунок 10 – Графічний інтерфейс розробленого додатку

Інтерфейс складається за таких частин:

1. Форми для введення значень M та N;
2. Кнопка для побудови нової моделі для введених M та N;
3. Кнопка для мінімізації базової моделі методом склейки;
4. Поле для виводу списку згенерованих функцій до та після мінімізації;
5. Графічне представлення моделі з логічними функціями (зліва) та стану моделі для заданого вектору помилок (справа);
6. Поле для відображення значення функції ребра;

7. Кнопка для забезпечення стійкості моделі для поточного вектору помилок;
8. Ряд перемикачів для маніпуляції значеннями індикаторних змінних вектору помилок.

Для створення нової моделі потрібно обрати ввести значення N, що відповідає за кількість індикаторних змінних в векторі помилок, та M, що відповідає за значення степеня відмовостійкості базової моделі. Наприклад, створимо модель, яка зберігатиме працездатність при відмові трьох з восьми модулів. Тобто, виберемо число 3 для значення M та число 8 для значення N. Після цього потрібно натиснути на кнопку Form. Результат показаний на рисунку 11.

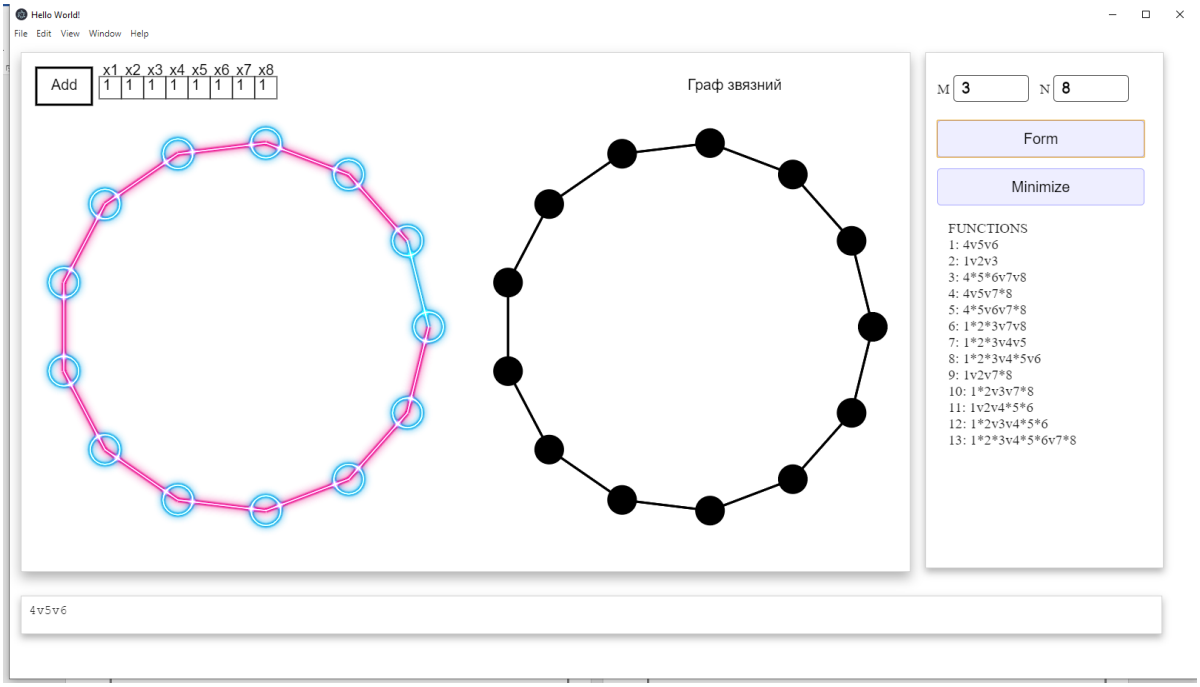


Рисунок 11 – Побудова базової моделі

Як можна побачити з результату роботи програми, було згенеровано 13 булевих функцій. Для перевірки правильності побудови моделі, виберемо довільний вектор відмови, який містить 4 (M + 1) нульові значення. Для зміни значення певної змінної на нульове, достатньо клікнути по ньому мишкою.

Аналогічним чином модна змінити нульове значення на одиницю. Результат показаний на рисунку 12.

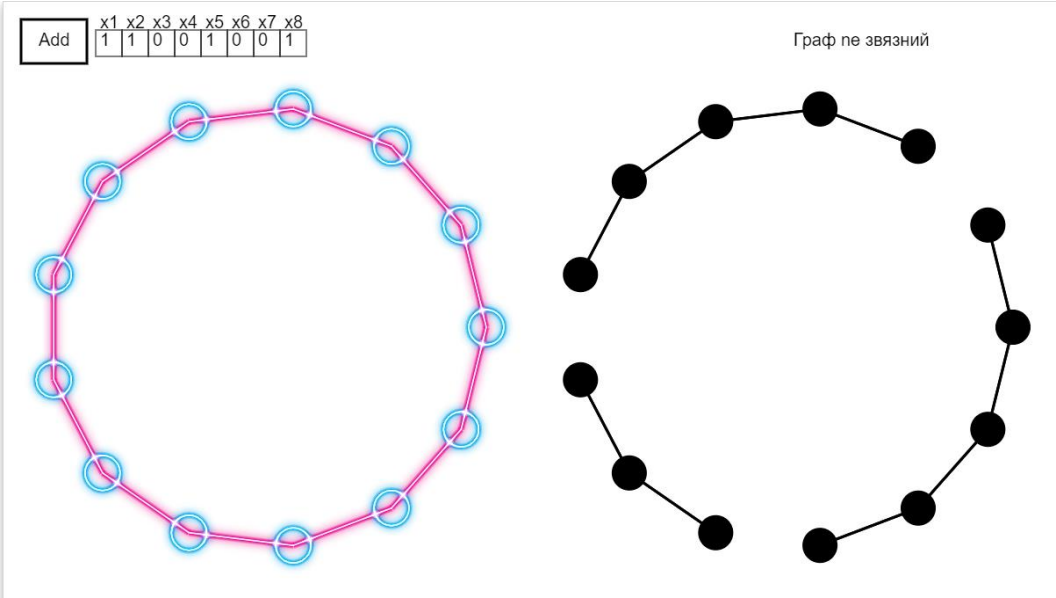


Рисунок 12 – Втрата зв’язності графом при встановленні в нуль M+1 змінних

Можна помітити, що граф втратив три ребра, проте для втрати зв’язності достатньо двох. Для проведення мінімізації моделі потрібно клікнути лівою кнопкою миші на кнопку Minimize. Результат показаний на рисунку 13.

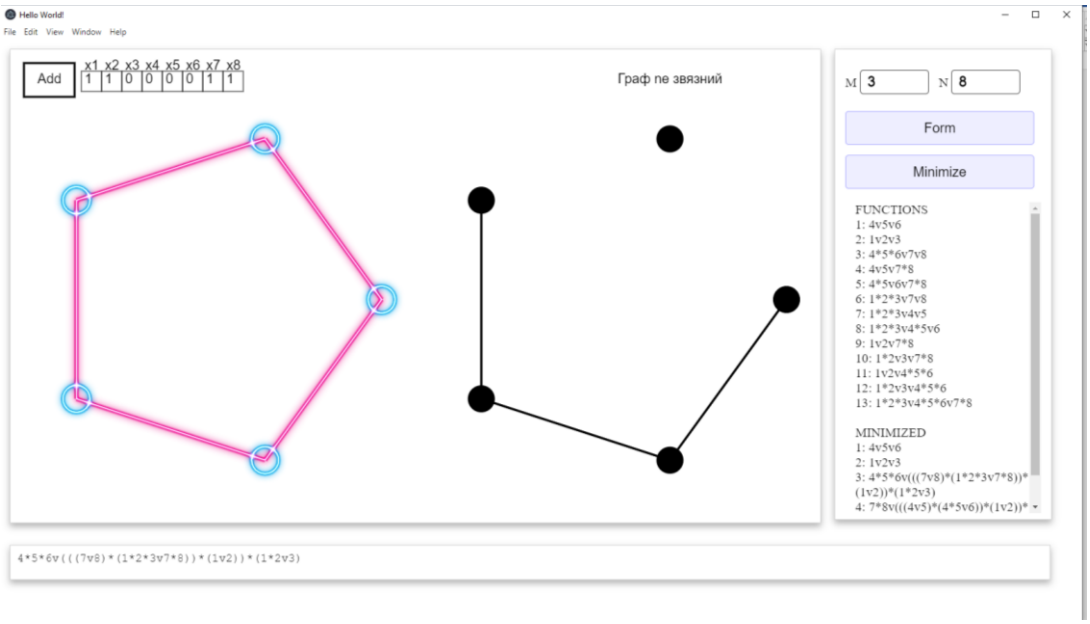


Рисунок 13 – Результат мінімізації моделі

Після мінімізації модель має всього 5 ребр, тобто набагато простішу структуру. На екрані також з’явився список функцій після склейок. Для перегляду значення функції конкретного ребра, потрібно навести на нього курсор мишки, тоді відповідне значення буде відображено в полі знизу, як показано на рисунку 14.

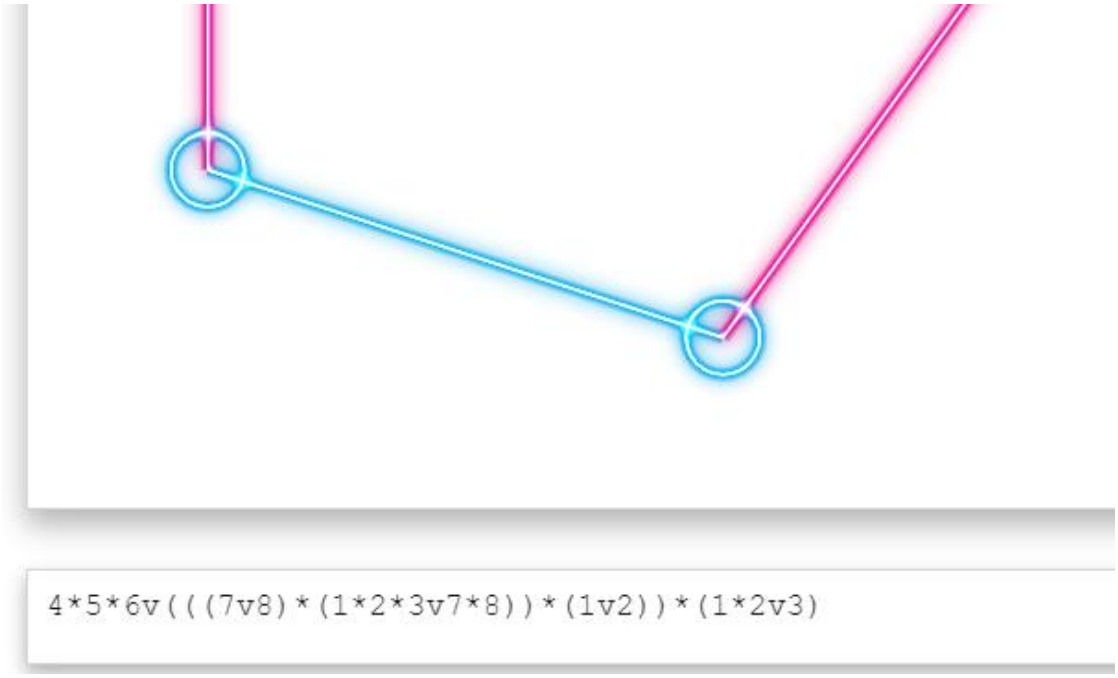


Рисунок 14 – Перегляд значення функції довільного ребра

Тепер можна додати моделі внутрішнє ребро для забезпечення відмовостійкості для розглянутого вектору помилок. Клікнувши на кнопку Add, що знаходиться біля перемикачів значень індикаторних змінних. Но моделі з’явиться додаткове ребро, також додатковий шлях з’явиться на графі. Результат показаний на рисунку 15.

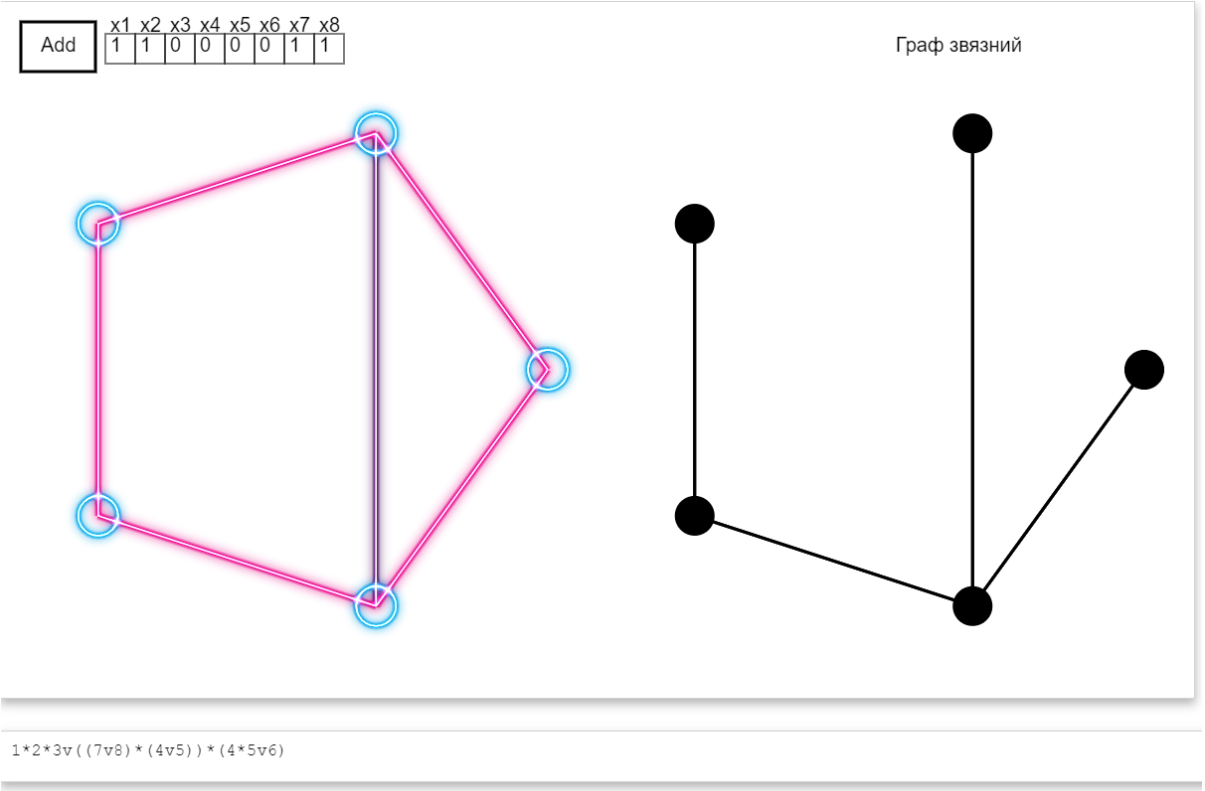


Рисунок 15 – Результат додавання внутрішнього ребра

# ВИСНОВОК

Метою цього дипломного проєкту є розробка програмного забезпечення для побудови та модифікації базових GL-моделей.

Проаналізувавши вимоги до програмної реалізації, було вирішено, що доцільно розробити систему у вигляді десктопного додатку з простим графічним інтерфейсом користувача. Також додаток може використовуватися на різних популярних платформах, зокрема Windows, Mac та Linux.

На основі аналізу найпопулярніших методів реалізації, було вирішено створити додаток за допомогою мови JavaScript, використовуючи фреймворк Electron.js для створення віконного додатку та використання технології 2D Canvas для відображення векторної графіки.

Розроблений додаток:

- має простий в користуванні графічний інтерфейс, адаптований для управління мишкою;
- дозволяє швидко генерувати реберні функції графо-логічної моделі;
- проводить мінімізацію моделі склеюванням ребр для спрощення структури моделі;
- показує реакцію моделі на введення довільного вектору помилок;
- дозволяє модифікувати модель методом проведення внутрішніх ребер.

Розробка програмного забезпечення виконана у повному обсязі та повністю відповідає поставленим вимогам до нього.

Тестування додатку виконано у відповідності до затвердженої програми та методики тестування.

Використання додатку дозволяє пришвидшити побудову GL-моделей та дозволяє проводити наглядні модифікації та тестування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Матвеевский В.Р. Надежность технических систем. Учебное пособие. М.: Московский государственный институт электроники и математики. 2002 г. 113 с.
2. Шкляр В.Н. Надёжность систем управления: учебное пособие. Томск: Издательство Томского политехнического университета. 2009 г. 126 с.
3. Галкин А.Г., Ковалев А.А. Основы теории надежности. Екатеринбург: Изд-во УрГУПС, 2014. 134 с.
4. Викторова В.С. Анализ надежности отказоустойчивых вычислительных систем. М.: ИПУ РАН. 2016. 117 с.
5. Федотов А.В., Н.Г. Скабкин. Основы теории надежности и технической диагностики. Омск: Изд-во ОмГТУ. 2010. 64 с.
6. Чебоксаров А.Н. Основы теории надежности и диагностика. Омск: СибАДИ. 2012. 76 с.
7. Горелик А.В. Ермакова О.П. Основы теории надежности в примерах и задачах. М.: МИИТ. 2009 г. 98 с.
8. Элементы теории надежности. *Научная билиотека*: URL: http://scask.ru/m_book_rdm.php?id=203 (дата звернення: 26.03.2020).
9. Reliability Block Diagrams Overview and Value. *Accendo Reliability:* URL: https://accendoreliability.com/reliability-block-diagrams-overview-and-value/ (дата звернення: 28.03.2020).
10. Intro to Fault Tree Analysis. *Accendo Reliability:* URL: https://accendoreliability.com/intro-to-fault-tree-analysis/ (дата звернення: 30.03.2020).
11. Benefits of Fault Tree Analysis. *Accendo Reliability:* URL: https://accendoreliability.com/benefits-of-fault-tree-analysis/ (дата звернення: 01.04.2020).

12. Markov Modeling for Reliability. *Mathpages.* URL: https://www.mathpages.com/home/kmath232/part1/part1.htm (дата звернення: 08.04.2020).
13. Markov Model Fundamentals. *Mathpages.* URL: https://www.mathpages.com/home/kmath232/part2/part2.htm
14. Petri Nets for System Reliability Modeling. *Accendo Reliability:* URL: https://accendoreliability.com/petri-nets-system-reliability-modeling/ (дата звернення: 04.04.2020).
15. Introduction to Physics of Failure Models. *Accendo Reliability:* URL: https://accendoreliability.com/introduction-physics-of-failure-models/ (дата звернення: 06.04.2020).
16. А.М. Романкевич, И.В. Майданюк, М.И. Хаитов. Об одном алгоритме формирования реберных функций графо-логических моделей отказоустойчивости многопроцессорных систем. *Радіоелектронні і комп'ютерні системи*. 2010. № 3. С. 56–61.
17. В.А. Романкевич, К.В. Морозов, А.П. Фесенюк. Об одном методе модификации реберных функций GL-моделей. *Радіоелектронні і комп'ютерні системи*. 2014. № 6. С. 95–99.
18. Qt features. *Qt.* URL: https://www.qt.io/product/features (дата звернення: 15.04.2020).
19. Java Swing Tutorial. *Java Point.* URL: https://www.javatpoint.com/java-swing (дата звернення: 15.04.2020).
20. Getting Started with NW.js. *NW.js.* URL: https://nwjs.readthedocs.io/en/latest/For%20Users/Getting%20Started/ (дата звернення: 15.04.2020).
21. What Is ElectronJS and Why Should You Use It?. *Alibaba Cloud.* URL: https://www.alibabacloud.com/blog/what-is-electronjs-and-why-should-you-use-it_581971 (дата звернення: 15.04.2020).

22. Karnaugh Maps, Truth Tables, and Boolean Expressions. *All about circuits.* URL: https://www.allaboutcircuits.com/textbook/digital/chpt-8/karnaugh-maps-truth-tables-boolean-expressions/ (дата звернення: 11.04.2020).
23. Minterm vs Maxterm Solution. *All about circuits.* URL: https://www.allaboutcircuits.com/textbook/digital/chpt-8/minterm-maxterm-solution/ (дата звернення: 12.04.2020).